НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК ___004.9_____

«До захисту допущено»
Завідувач кафедри СПСКС

_____ Віталій РОМАНКЕВИЧ_

"___"_____20___р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
(Комп'ютерні системи та компоненти)

на тему: Спосіб обробки інформації, представленої природомовними об'єктами _____
_____

Виконав: студент ІІ курсу, групи _КВ-91_мп
Здирко Владислав Володимирович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник Орлова Марія Миколаївна, доцент кафедри СПСКС

к.т.н., доцент _____ _____

Консультант з нормоконтролю доцент, с.н.с.,к.т.н. Юлія БОЯРІНОВА _ _____

Рецензент_____ _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
за освітньо-професійною програмою
Спеціальність 123 Комп'ютерна інженерія
Комп'ютерні системи та компоненти

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС

_____ _Віталій РОМАНКЕВИЧ__

«_01_»_____12_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Здирко Владислав Володимирович
_____

1. Тема дисертації Спосіб обробки інформації, представленої природомовними об'єктами
_____
_____,
науковий керівник дисертації Орлова Марія Миколаївна, доцент кафедри СПСКС к.т.н. доцент _____,

затверджені наказом по університету від «_12__» _11_ 2020 р. № 3298-С

2. Термін подання студентом дисертації 10 грудня 2020 р.
_____

3. Об'єкт: способи обробки текстів методами машинного навчання.
_____
_____

4. Предмет дослідження є є способи порівняння текстових данизх та способи узагальнення текстових даних. Виділення ключових слів для створення текстових-векторів.

5. Перелік завдань, які потрібно розробити: аналіз існуючих методів виділення кочових фрагментів, виділення основних рис з існуючих алгоритмів, розробка нового алгоритму виділення ключових фрагментів, програмна реалізація розробленого алгоритму

6. Перелік ілюстративного матеріалу: презентація

7. Перелік публікацій За тематикою проведених досліджень повинно бути опубліковано 2 наукові праці, а саме 2 тези доповідей на конференціях.

8. Дата видачі завдання 5 вересня 2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Формування мети та цілі роботи	01.11.2019	
2	Дослідження теоретичного матеріалу	01.03.2020	
3	Дослідження існуючих алгоритмів	01.05.2020	
4	Розробка нового алгоритму	01.07.2020	
5	Реалізація нового алгоритму	01.09.2020	
6	Проведення аналізу роботи алгоритму	01.10.2020	
7	Написання дисертації	20.11.2020	
8	Попередній розгляд магістерської дисертації на кафедрі	26.11.2020	

Студент_____ _____ Владислав ЗДИРКО

Науковий керівник дисертації _____ Марія ОРЛОВА

РЕФЕРАТ

Актуальність теми

Задача обробки інформації, яка представлена природномовною формою, актуальна з часів виникнення писемності. Такі проблеми, як коректний переклад, пошук інформації, класифікація текстів постійно супроводжували людство з цих часів. Після появи доступних персональних комп'ютерів (ПК) обсяг інформації в світі перевищив всі прогнози. Тому не дивно, що постала гостра необхідність в вирішенні перелічених вище задач програмними методами.

На сьогодні контроль за поширенням і доступністю інформації, контроль за її цілісністю, унікальністю та самобутністю також дуже необхідний. Існують цілі галузі, де перевірка та захист інформації вкрай необхідні. До таких областей відносяться наука (патентування, наукові статті та інше) та культура (авторське право та інше).

Для вирішення проблем унікальності, наявності запозичень та іншого подібного було розроблено цілу низку статистичних та програмних методів. На їх основі базуються такі сервіси як Unicheck та інші. Але вони не в повному обсязі вирішують поставлену задачу, оскільки на сьогодні ці сервіси не є чутливими до заміни слів на синоніми, антоніми, зміну мови (переклад) та заміну частини символів.

Мета роботи: Мета роботи полягає в покращення методів обробки текстових даних. Для покращеного розпізнавання плагіату, незареєстрованих запозичень. А також для покращення системи пошуку текстової інформації.

Для досягнення поставленої мети в даній роботі вирішуються наступні задачі.

1. Аналіз способів класифікації інформації, яка представлена в природномовній формі.

2. Аналіз способів формування векторних представлень інформації в природомовній формі.
3. Аналіз методів продовження (угадування) інформації в природомовній формі з використанням машинного навчання.
4. Дослідження та порівняння класичних статистичних методів та машинного навчання в NLP задачах.
5. Підвищення ефективності існуючих методів класифікації за рахунок розробки модифікованого способу класифікації інформації в природомовній формі та їх порівняння методами машинного навчання.

Об'єктом дослідження є способи обробки текстів, методами машинного навчання.

Предметом дослідження є способи порівняння текстових даних та способи узагальнення текстових даних. Виділення ключових слів для створення текстових-векторів.

Методи дослідження. В роботі використовуються методи штучного інтелекту абстракції, класифікації, порівняння природомовної форми інформації Також у роботі використовуються методи взаємодії штучного інтелекту і класичних статистичних методів.

Наукова новизна одержаних результатів полягає в наступному:

1. Проаналізовано основні способи класифікації та порівняння інформації, яка представлена в природномовній формі.
2. Запропоновано спосіб підвищення ефективності методу порівняння та представлення інформації в природномовниій формі.
3. Досліджено та запропоновано впровадження даних методик в сфери захисту інтелектуальної власності.

Практична цінність одержаних результатів

В даній роботі запропоновано альтернативні методи з перевірки та обробки тексту, які забезпечують захист, обробку інформації, представленої в текстовій формі.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на:

- XIII науковій конференції молодих вчених «Прикладна математика та комп'ютинг» ПМК-2020;
- VII Міжнародної науково-технічної конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами».

Публікації. За тематикою проведених досліджень опубліковано 2 наукові праці, а саме тези доповідей на 2-х конференціях.

Структура та обсяг роботи.

Магістерська дисертація складається зі вступу, чотирьох розділів, висновків по кожному розділу та загальних висновків по роботі в цілому, списку використаних літературних джерел (16 найменувань). Повний обсяг дисертації – 88 сторінок, у тому числі 78 сторінок основного тексту, 25 рисунків, 7 таблиць.

У вступі надано загальну характеристику проблем задач обробки натуральних мов, сформовано мету досліджень, а також сформульовано практичну цінність роботи.

У першому розділі була сформована задача даної роботи, а також розглянуті базові принципи обробки природної мови (Natural language processing) та сучасні методи використання машинного навчання та нейронних мереж для обробки природної мови.

У другому розділі розглянуто актуальні методики з вирішення проблеми обробки натуральних мов (Natural language processing) та сучасні методи використання машинного навчання та нейронних мереж для обробки природної мови. Розглянуто та проаналізовано проблеми векторного представлення природомовних об’єктів інформації, продовження (угадування) природомовної форми інформації з використанням машинного навчання. Показано, що використання векторних представлень слів у обробці природних мов дає велику перевагу перед більш простими методами, як мішок слів, і дозволяє знаходити додаткові, неочевидні взаємозв’язки між текстами. Проведено порівняння та аналіз існуючих рішень, а також обґрунтовано вибір інструментарію для вирішення поставлених задач.

У третьому розділі розроблено та описано програмний комплекс для вирішення визначених задач обробки натуральних мов, представлена порівняльна характеристика запропонованого способу з вже існуючими.

У висновках було проаналізовано отриманий результат.

Ключові слова: Обробка натуральних мов, інтерпретація мовних структур, класифікація тексту, методи генерації тексту, NLP, машинне навчання.

ABSTRACT

Actuality of theme

The task of information processing, which is presented in natural language form, is relevant since the days of writing. Problems such as correct translation, search for information, classification of texts have constantly accompanied mankind since then. With the advent of affordable personal computers (PCs), the world's information has exceeded all expectations. Therefore, it is not surprising that there is an urgent need to solve the above problems by software methods.

Today, control over the dissemination and availability of information, control over its integrity, uniqueness and identity is also very necessary. There are whole areas where information verification and protection is essential. Such areas include science (patents, scientific articles, etc.) and culture (copyright, etc.).

A number of statistical and software methods have been developed to address uniqueness, borrowing, and the like. Services such as Unicheck and others are based on them. But they do not fully solve the problem, because today these services are not sensitive to the replacement of words with synonyms, antonyms, change of language (translation) and replacement of some characters.

Purpose: The purpose of the work is to improve the methods of processing text data. For improved recognition of plagiarism, unregistered borrowings. And also to improve the text search system.

To achieve this goal in this work the following tasks are solved.

1. Analysis of ways to classify information that is presented in natural language form.
2. Analysis of ways of forming vector representations of information in natural language form.
3. Analysis of methods of continuation (guessing) of information in natural language form using machine learning.
4. Research and comparison of classical statistical methods and machine learning in NLP problems.

5. Improving the efficiency of existing classification methods by developing a modified method of classifying information in natural language form and comparing them with machine learning methods.

The object of research is the methods of word processing, methods of machine learning.

The subject of the research is the ways of comparing textual data and ways of generalizing textual data. Highlight keywords to create text vectors.

Research methods. The methods of artificial intelligence of abstraction, classification, comparison of natural form of information are used in the work. The methods of interaction of artificial intelligence and classical statistical methods are also used in the work.

The scientific novelty of the obtained results is as follows:

1. The main methods of classification and comparison of information presented in natural language form are analyzed.

2. The way of increase of efficiency of a method of comparison and representation of the information in a natural language form is offered.

3. The introduction of these methods in the field of intellectual property protection is researched and offered.

The practical value of the results obtained

This paper proposes alternative methods for checking and processing text, which provide protection, processing of information presented in text form.

Approbation of work. The main provisions and results of the work were presented and discussed at:

- the XIII Scientific Conference of Young Scientists "Applied Mathematics and Computing" PMK-2020;
- the VII International Scientific and Technical Conference "Modern methods, information, software and hardware management organizational and technical and technological complexes ". .

Publications. Two scientific papers were published on the subject of the conducted researches, namely abstracts of reports at 2 conferences.

Structure and scope of work

The master's dissertation consists of an introduction, four chapters, conclusions on each section and general conclusions on the work as a whole, a list of used literature sources (16 titles). The full volume of the dissertation is 87 pages, including 78 pages of the main text, 2518 figures, 7 tables.

In the introduction the general characteristic of problems of problems of processing of natural languages is given, the purpose of researches is formed, and also the practical value of work is formulated.

In the first section the task of the given work was formed, and also the basic principles of processing of natural language (Natural language processing) and modern methods of use of machine learning and neural networks for processing of natural language are considered.

The second section discusses current methods for solving the problem of natural language processing (Natural language processing) and modern methods of using machine learning and neural networks for natural language processing. The problems of vector representation of natural language objects of information, continuation (guessing) of natural language form of information with the use of machine learning are considered and analyzed. It has been shown that the use of vector representations of words in natural language processing has a great advantage over simpler methods, such as a bag of words, and allows you to find additional, non-obvious relationships between texts. The comparison and analysis of existing solutions are carried out, and also the choice of tools for the decision of the set tasks is substantiated.

In the third section the software complex for the decision of the certain problems of processing of natural languages is developed and described, the comparative characteristic of the offered way with already existing is presented.

The conclusions analyzed the obtained result.

Keywords: Natural language processing, interpretation of language structures, text classification, text generation methods, NLP, machine learning.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

RNN - recurrent neural networks*, RNN* це клас штучних нейронних мереж, у якому з'єднання між вузлами утворюють граф орієнтований у часі. Таким чином вони зберігають послідовність входів даних для кращої їх обробки.

NLP - Natural Language Processing. Загальна направлення штучного інтелекту та математичної лінгвістики. Вивчає проблеми компютерного аналіза мов та синтезу натуральних мов.

Word embeddings - загальна назва для різних підходів до моделювання мови, спрямованих на зіставлення слів з деяким набором(словником) векторів.

k-NN – метод k-найближчих сусідів.

LR – Logistic Regression – логістична регресія.

ВСТУП

Задачі обробки інформації, які представлені в природномовній формі, актуальна з часів започаткування письма. Такі проблеми як коректний переклад, пошук інформації, класифікація текстів постійно супроводжували людство з виникненням писемності. Після виникнення доступних персональних комп'ютерів (ПК) збільшення обсягу інформації в світі перевищило всі прогнози. Тому не дивно, що постала гостра необхідність в вирішенні перелічених вище задач програмними методами.

На сьогодні поширення і доступність інформації, контроль за цілісністю унікальністю та самобутністю також дуже необхідний. Існують цілі галузі, де перевірка та захист інформації вкрай необхідні. Такими галузями являються: наука (патентування, наукові статті та інше) та культура (авторське право та інше).

Для вирішення проблем унікальності, наявності запозичень тощо було створено цілу низку статистичних і програмних методів. На їх основі базуються такі сервіси як Unicheck та інші. Але вони не в повному обсязі вирішують дану задачу, оскільки на теперішній час такі сервіси не є чутливими до заміни слів на синоніми, антоніми, зміну мови (переклад) та заміну частини символів.

В даній роботі запропоновано альтернативні методи з перевірки та обробки тексту, які призвані забезпечити захист і обробку інформації, представленої в текстовій формі.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

1.1 Постановка задачі обробки природних мов

Розглянемо спочатку особливості обробки природних мов NLP (Natural Language Processing). Сучасна методологія обробки текстової інформації представлена двома варіантами: це статистичні методи обробки інформації та методи машинного навчання (штучного інтелекту). Штучний інтелект — це системи, які здатні на самостійну інтерпретацію даних, та подальше використання набутих навичок для покращення процесу обробки (самонавчання).

NLP як наука має на меті інтерпретацію природомовних об’єктів в об’єкти, які є простими і зрозумілими для обробки комп’ютером В основі цього лежать такі проблеми як машинний переклад, розпізнавання мов, передбачення слів, діалогові системи, розпізнавання іменованих об’єктів, пошук інформації, класифікація тексту, узагальнення інформації (text summarization) та багато іншого. Тому справедливо буде підсумувати, що область NLP охоплює всі взаємодії між комп'ютером і людиною з використанням письмової або усної природної мови. Дослідження, пов’язані з маніпулюванням і розумінням природних мов комп’ютером, є на сьогодні особливо актуальні. Обробка мов базується на розумінні таких частин мови, як фонетика, морфологія, синтаксис, семантика.

Фонетика і фонологія є особливо важливими аспектами в розпізнаванні мови при перетворенні звуків в реальні слова, які можуть бути оброблені комп'ютером. Морфологія стосується значення і архітектури слів. Стемінг і лематизації засновані на цьому компоненті шляхом перетворення слів, таких як «going», до слова «go». Порядок слів і побудова граматичних правильних пропозицій досліджується синтаксисом. На відміну від цього, семантика досліджує значення пропозицій, які побудовані з використанням синтаксису і

морфологічних форм слова. Щоб отримати передбачуваний загальний зміст повідомлення, прагматика використовує контекст ситуації. Тому комп'ютер повинен враховувати всі частини природної мови, щоб використовувати його.

Щоб дійти до вирішення поставлених NLP задач потрібно обробляти всі ці параметри. Формулювання завдань є не дуже складним, проте самі завдання зовсім не є простими, оскільки повинне оброблятися текст, представлений природною мовою. Явища полісемії (багатозначні слова, які мають загальний вихідний сенс) і омонімії (різні за змістом слова, що вимовляються і пишуться однаково) характерні для будь-якої природної мови. І якщо носій української мови добре розуміє, що теплий прийом має мало спільного з бойовим прийомом, з одного боку, і теплим пивом, з іншого, автоматичній системі доводиться довго цьому вчитися.

На відміну від обробки зображень, по NLP досі можна зустріти статті, де описуються рішення, які використовують не машинен навчання, а класичні алгоритми типу SVM або Xgboost, і показують результати, які не дуже сильно поступаються. Тим не менше, кілька років тому нейромережі почали перемагати класичні моделі. Важливо відзначити, що для більшості завдань рішення на основі класичних методів були унікальні, як правило, не схожі на вирішення інших завдань як за архітектурою, так і за тим, як відбувається збір і обробка ознак.

Однак нейромережеві архітектури набагато більш узагальнені. Архітектура самої мережі, швидше за все, теж відрізняється, але набагато менше, йде тенденція в бік повної універсалізації.

1.2 Попередня обробка тексту

Текст представляє собою послідовність слів, які зв’язані між собою контекстом. Проте розглядати слова як набір символів не є дуже доцільним, оскільки контекст символів в переважній більшості моделей не мають окремого контексту (є винятки в різних мовах, так, наприклад, в українській

мові зміна символів кінця слова представляє собою зміну відмінка, тобто характер дії змінюється).

Для того, щоб мати змогу вирішувати поставленні задачі, прийнято збирати текст, який необхідний для навчання алгоритмів в корпуси текстів. Текстовий корпус являє собою найбільш повну збірку текстової інформації, яка повинна охоплювати якомога більшу інформаційну область. Далі після первинного навчання прийнято впроваджувати додаткове навчання на базі спеціалізованих корпусів. Це дозволить найбільш точно налаштовувати модель, що покращить вирішення поставлених задач. Але для цього необхідно виконати процес очищення даних. Розглянемо основні етапи очищення даних з поясненнями, які представлено на наступних підрозділах.

1.3 Токенізація

Коли мова йде про мову, основна одиниця, з якою працює користувач, це слово, або більш формально «токен». Цей термін використовується, тому що не дуже зрозуміло, чи є словом будь-яке числове значення, наприклад, 2128506. Відповідь на це не є очевидною. Токен, зазвичай, відділений від інших токенів пробілами або знаками пунктуації. І як зрозуміло зі складностей, які описано вище, дуже важливий контекст кожного токена. Є різні підходи, але в 95 % випадків таким контекстом, який розглядається при роботі моделі, виступає речення, що включає вихідний токен (рисунок 1.1.).

Рисунок 1.1 – Результат роботи простого токенізатора

Токенізація розбиває тексти, що знаходяться в обробці, на короткі текстові об'єкти і є найпершою задачею в будь-якому циклі попередньої обробки тексту. Крім розбиття на маленькі блоки, цілі речення також можуть бути результатом токенізації. Простий токенізатор слів може бути реалізований багатьма мовами. Цей простий базовий підхід має декілька недоліків через відсутність ідентифікуючих слів, які семантично пов'язані один з одним. Однак простий токенізатор ділить будь-яку фразу на деякі маркери.

Використовуючи маркери, можна створити так звані *n*-грами, які вказують набір токенів з довжиною *n*. «Грама» - це грецьке слово, що позначає букву або знак. Коли мова йде про набір з *n* букв у словах, то вважається, що мова йде про символи з *n* грамів.

1.4 Видалення стоп-слів

Важливою частиною обробки тексту є видалення слів, які не містять інформації (або майже її не містять), і представляють собою допоміжні слова, дієслова, артикули тощо. У більшості мов такі слова існують, і існують декілька алгоритмів машинного навчання, які використовують ці слова. В англійській мові такими словами можуть бути «the», «a» або «an». Тому прийнято формулювати список стоп-слів, і при обробці тексту видаляти їх. Ці списки доступні в літературі і часто реалізуються в різних програмних пакетах.

1.5 Стемінг

Стемінг (*stemming*) - це процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення та суфікс. Результати стемінгу іноді дуже схожі на визначення кореня слова, але його алгоритми базуються на інших принципах. Тому слово після обробки алгоритмом стемінгу (стематизації) може відрізнятися від морфологічного кореня слова. Стемінг застосовується в лінгвістичній морфології та в інформаційному

пошуку. Багато пошукових систем використовують стемінг для об’єднання слів, у яких збігаються форми після стематизації, вони вважають такі слова синонімами. Цей процес називають злиттям. Комп’ютерна програма, що реалізує алгоритм стемінгу, іноді має назву стемер.

Перший алгоритм визначення основ, заснований на видаленні найдовших суфіксів і правописі, був розроблений в 1968 році [8]. До теперішнього моменту алгоритм визначення походження портеру являє собою сучасний підхід і видаляє всі суфікси зі слів для збереження основи слова. Хоча цей метод добре працює на англійській мові, у німецької мови є деякі недоліки, пов'язані з тим, що німецькі слова, зазвичай, не створюються шляхом додавання суфіксів

1.6 Лематизація

Лематизація - це процес зіставлення кожного слова в тексті до їх базової форми. Дієслова перетворюються в свою початкову форму, іменник відновлюється в однині, а прислівники або прикметники передбачають їх позитивний формат. Цей метод базується на морфологічному аналізі і часто використовує словник, наприклад, WordNet, де можна знайти лему для кожного зміненого слова. Цей етап попередньої обробки скорочує векторний простір шляхом зіставлення різних форм слів з їхнім спільним представленням. Оскільки лематизація підтримується словниковими значенням, вона може зіставити «best» з його лемою «good».

1.7 Представлення слів в векторній формі

Word embedding.Сам по суті embedding - це зіставлення в довільній суті (наприклад, вузла в графі або чпстини картинки) деякого вектору. По своїй суті ідея дуже проста і продуктивна, ряд натуральних чисел безкінечний і можна найменувати усі слова числом без будь-яких проблем.

Але у цієї ідеї є і істотний недолік: слова в словнику слідують в алфавітному порядку, і при додаванні слова потрібно перенумеровувати заново більшу частину слів. Але навіть це не є настільки важливим, а дуже важливим є те, що буквене написання слова ніяк не пов'язане з його змістом (цю гіпотезу ще в кінці XIX століття висловив відомий лінгвіст Фердинанд де Соссюр) [2]. Справді слова "півень", "курка" і "курча" мають дуже мало спільного між собою і стоять в словнику далеко один від одного, хоча очевидно позначають різні об’єкти одного й того ж виду птиці. Тобто можна виділити два види близькості слів: лексичний і семантичний. Як можна побачити на прикладі з куркою, ці близькості не обов'язково збігаються.

Щоб отримати можливість представити семантичну близькість, було запропоновано використовувати embedding, тобто призначати словам якийсь вектор, що відображає його значення в "просторі смислів".

Найпростіший спосіб представлення, пов’язаний з тим, що природно буде взяти вектор довжини будь-якого словника і поставити тільки одну одиницю в позиції, що відповідає номеру слова в словнику. Цей підхід називається one-hot encoding (OHE). OHE все ще не має властивості семантичної близькості [6].

Значення одного слова може бути і не дуже важливим, оскільки мова (і усна, і письмова) складається з наборів слів, які називаються текстами. І тому якщо необхідно якось уявити тексти, то береться OHE-вектор кожного слова в тексті і складається разом. Тобто на виході отримується просто підрахунок кількості різних слів у тексті в одному векторі. Такий підхід називається "мішком слів" (bag of words, BoW) [11], тому що ми втрачаємо всю інформацію про взаємне розташування слів у тексті.

При використанні цього метода виникає ряд проблем. Проте вже можна співставляти корпуси текстів.

Але цього виявилось не достатньо, оскільки багато семантичної інформації зберігається у вигляді послідовності слів. Тому прийнято

представляти (набір текстів) у вигляді матриці "слово-документ" (term-document) [11]. Варто зазначити, що в області інформаційного пошуку (information retrieval) ця матриця має назву "зворотного індексу" (inverted index), в тому сенсі, що звичайний/прямий індекс виглядає як "документ-слово" і дуже незручний для швидкого пошуку.

Ця матриця призводить до тематичних моделей, де матрицю "слово-документ" намагаються представити у вигляді добутку двох матриць "слово-тема" і "тема-документ". У найпростішому випадку береться така матриця і за допомогою SVD-розкладання одержується уявлення слів через теми і документів через теми (рисунок 1.2).

$$\begin{matrix} & X & & U & & \Sigma & & V^T \\ & (\mathbf{d}_j) & & & & & & (\hat{\mathbf{d}}_j) \\ & \downarrow & & & & & & \downarrow \end{matrix}$$

$$(\mathbf{t}_i^T) \rightarrow \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} = (\hat{\mathbf{t}}_i^T) \rightarrow \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} \dots \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \cdot \begin{bmatrix} [& \mathbf{v}_1 &] \\ & \vdots & \\ [& \mathbf{v}_l &] \end{bmatrix}$$

Рисунок 1.2- SVD-розкладання

Описані вище підходи були (і залишаються) гарними для часів (або областей), де кількість текстів незначна і словник обмежений, хоча, як можна побачити, там теж є свої складності. Але з появою мережі Інтернет і інших мережних технологій все стало одночасно і складніше, і простіше: в доступі з'явилося велика кількість текстів, і ці тексти використовують змінні словники, які розширюються. З такою проблемою і з таким обсягом текстів раніше відомі моделі не могли впоратися. Кількість слів в англійській мові приблизно становить мільйон - матриця спільних тільки пар слів буде 10 ^ 6 x 10 ^ 6. Така матриця навіть зараз не може бути записана в пам'ять комп'ютерів, а, скажімо, 10 років тому про таке можна було не мріяти. Звичайно, було розроблено багато

способів, що спрощують або розпаралелюють обробку таких матриць, але все це були паліативні методи.

Тому в 2013 році чеськиі аспірант Томаш Міколов [11] запропонував свій підхід до word embedding, який він назвав word2vec.

Модель, запропонована Міколовим, дуже проста (і тому так хороша) - можна передбачати ймовірність слова за його оточенням (контекстом). Тобто можна будемо вчити такі вектора слів, щоб ймовірність, що привласнюється слову моделлю, була близька до ймовірності знайти це слово в цьому оточенні в реальному тексті формула нище.

$$P(w_o|w_c) = \frac{e^{s(w_0,w_c)}}{\sum w_i \in \forall e^{s(w_i,w_c)}} \quad ,$$

де: ω(o) - вектор цільового слова, ω(c) - певний вектор контексту, обчислений (наприклад, шляхом усереднення) з векторів, що оточують потрібне слово іншими словами, s(ω(1), ω(2)) - функція, яка зіставляє одне число з двох векторів (наприклад, це може бути згадуване вище метод косинусни відстаней).

Наведена формула називається softmax, тобто "м'який максимум", м'який - в сенсі диференційований. Це потрібно для того, щоб модель могла навчитися за допомогою backpropagation, тобто процесу зворотного поширення помилки.

Процес тренування влаштований таким чином: береться послідовно $(2k + 1)$ слів, слово в центрі є тим словом, яке повинно бути передбачене. А навколишні (близькі за розташуванням) слова є контекстом довжини по k з кожного боку. Кожному слову в моделі підтверджено унікальний вектор, який змінюється в процесі навчання даної моделі.

1.8 Логістична регресія

Логістична регресія (англ. logistic regression) - статистичний регресійний метод. Логістичну регресію дослідники застосовують у задачах класифікації,

тобто у випадку, коли залежна змінна представляється різними категоріями, тобто може набувати тільки скінченної множини значень (наприклад, 0 та 1). Якщо порівнювати зі звичайною регресійною моделлю, то метод логістичної регресії не виконує прогноз для значень числової змінної виходячи з вибірки вихідних значень незалежної змінної. Замість цього, значеннями для функції є ймовірність того, що дане вихідне значення належить до одного із класів. Аби спростити задачу припустимо, що у нас є тільки два класу і ймовірність, яку ми будемо визначати, вірогідності того, що певне значення належить класу "+". Таким чином, результат логістичної регресії завжди буде знаходитись в інтервалі [0, 1], як і ймовірність. Головна суть логістичної регресії полягає в тому, що простір вихідних значень незалежної змінної може бути розділений лінійною площиною (тобто, прямою) на два класи. Якщо точніше, то під лінійною площиною при двох параметрах (два виміри) мають на увазі пряму лінію, що розділяє класи (без вигинів). У випадку трьох вимірів — це площини, і так далі [5]. Ця межа визначається в залежності від наявних вихідних даних та навчального алгоритму. Щоб все працювало, точки вихідних даних повинні розділятися лінійною площиною на дві вищезазначені області. Якщо точки вихідних даних задовольняють цій вимозі, то їх можна назвати лінійно роздільними, як це представлено на рисунку 1.3.

Зазначена на рисунку площина називається лінійним дискримінантом. Вона є лінійною з точки зору своєї функції і моделі, а також проводить поділ, якщо точніше то дискримінацію точок на різні класи. Якщо неможливо провести лінійний розподіл точок у вихідному просторі або вони взагалі лінійно нероздільні, то варто спробувати перетворити вектори ознак в простір більшої розмірності, додавши додаткові ефекти взаємодії, члени більш високого ступеня та інше. Такий метод називають ”трюк з ядром” (Kernel trick). Використання лінійного алгоритму в такому випадку дає певні переваги для навчання нелінійної функції, оскільки межа стає нелінійною при поверненні у

вихідний простір [7]. Визначення логістичної моделі: нехай є деяка випадкова величина Y, що може набувати лише двох значень.

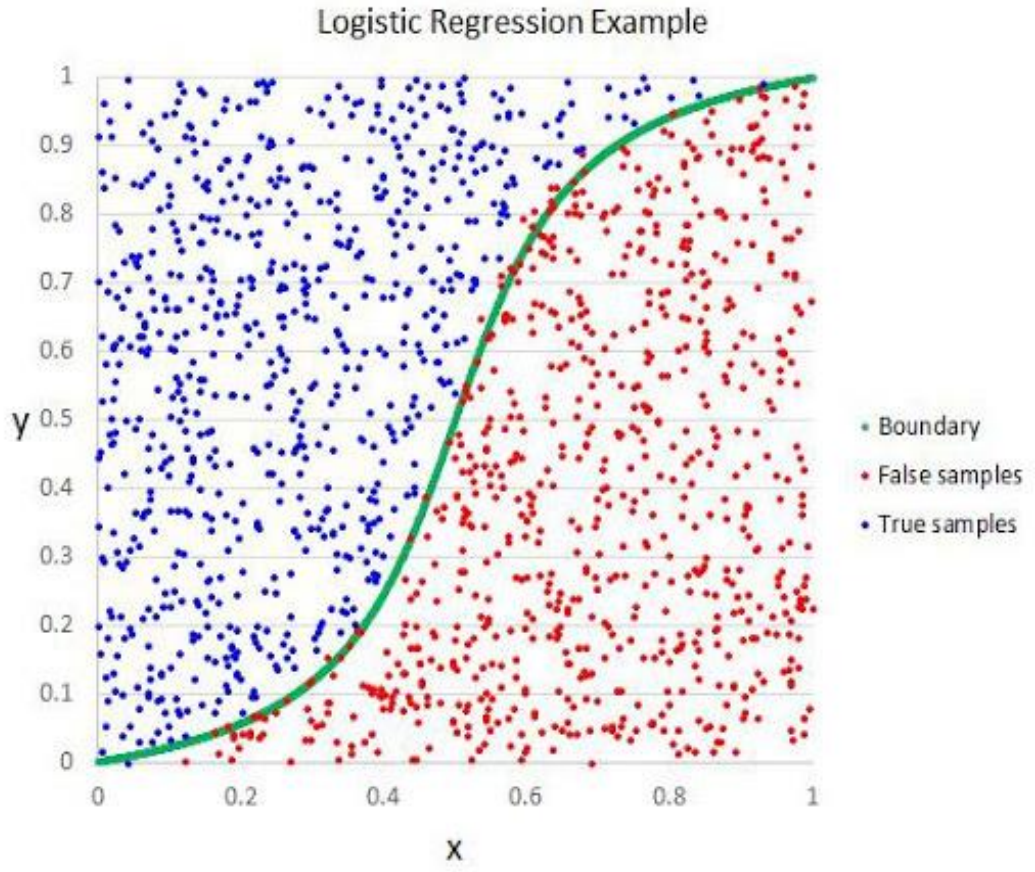

Рисунок 1.3 – Лінійно роздільний набір даних

Ці значення позначаються цифрами 0 і 1 (2 класи). Припустимо, що ця величина залежить від деякої множини незалежних змінних $x = (1, x1, \dots, xn)$ T . Залежність Y від $x1, \dots, xn$ можна визначити, ввівши додаткову змінну y*, де y*= $\theta\ T = \theta 0 + \theta 1 x1 + \dots + \theta n xn + \varepsilon$ Тоді:

$$Y = \begin{cases} 0, & y^* \leqslant 0 \\ 1, & y^* > 0 \end{cases}$$

При визначенні логістичної моделі стохастичний доданок ε вважається випадковою величиною з логістичним розподілом ймовірностей. Тому можна зробити висновок, що для певних конкретних значень змінних $x* = x1*, \dots,$

xn $*$ одержується відповідне значення y* і ймовірність того, що Y=1 є наступною:

$$p(Y=1)=p(y^*>0)=p(\theta^T x^*+\varepsilon>0)=p(\varepsilon>-\theta^T x^*)=p(\varepsilon\leqslant\theta^T x^*)=\Lambda(\theta^T x^*).$$

Передостання рівність випливає з симетричності логістичного розподілу, Λ позначає логістичну функцію — функцію розподілу логістичного розподілу:

$$A(x)=\frac{e^x}{1+e^x}=\frac{1}{1+e^{-x}}$$

Тому можна сказати, що для конкретного значення $x\ i$ випадкова величина $Y\ i$ має розподіл Бернуллі: $Y\ i\sim B(1,\ \Lambda(\theta\ Tx\ i\ ))$ Логістична модель задовольняє наступній умові [6]:

$$ln(x)\frac{p(1|X)}{1-p(1|X)}=ln(x)\frac{p(1|X)}{p(0|X)}=b_0+b_1x_1+b_jx_j$$

1.9 Наївний Баєсів класифікатор

У машинному навчанні наївні класи Бейєса є сімейством простих імовірнісних класифікаторів, заснованих на застосуванні теореми Байєса з сильними (наївними) припущеннями про незалежність між функціями [9]. Наївний Байєс широко вивчався з 1950-х років. Він був введений під іншим іменем в текстовий пошук для спільноти на початку 1960-х років і залишається популярним (базовим) методом для класифікації тексту, вирішує проблему розгляду документів як однієї категорії (наприклад, спам, новини, спорт або політика тощо), тобто класифікації з частотою слів як функції. За умови відповідної попередньої обробки, цей метод є конкурентоспроможним у цьому застосуванні за допомогою більш розвинутих, сучасних методів, включаючи метод опорних векторів[11]. Він також знаходить застосування в автоматичній медичній діагностиці. Класифікатори наївного Байєса мають високу масштабованість, що потребує ряду параметрів, лінійних за кількістю змінних

(функцій/предикторів) у проблемі навчання. Тренування максимальної ймовірності можна зробити, оцінюючи вираз із замкнутою формою, який приймає лінійний час, а не дорогим ітераційним наближенням, як це використовується для багатьох інших типів класифікаторів. У статистиці та комп'ютерній літературі наївні моделі Байєса відомі під різними назвами, в тому числі простими Байєсами та незалежними Байєсами [12]. Всі ці імена вказують на використання теореми Байєса в правилі рішення класифікатора, але наївний Байес не є (обов'язково) байєсівський методом. Для деяких типів моделей вірогідності наївні класифікатори Байєса можуть бути дуже ефективними. У багатьох практичних додатках оцінка параметрів для наївних моделей Байеса використовує метод максимального правдоподібності: іншими словами, можна працювати з наївною моделлю Байєса, не приймаючи вірогідність Байєса або використовуючи будь-які байєсівські методи. Незважаючи на наївний дизайн і, мабуть, надто спрощені припущення, наївні класифікатори Байєса працювали досить добре в багатьох складних реальних ситуаціях [13]. У 2004 році аналіз проблеми класифікації Байєса показав, що існують обґрунтовані теоретичні причини для очевидної неімовірнісної ефективності наївних класифікаторів Байєса. Проте всеосяжне порівняння з іншими алгоритмами класифікації в 2006 році показало, що класифікація Байєса випереджає інші підходи, такі як дерева рішень або випадкові ліси. Перевага наївного Байеса полягає в тому, що потрібна лише невелика кількість навчальних даних для оцінки параметрів, необхідних для класифікації.

1.10 Нейронні мережі прямого поширення

Нейронна мережа прямого поширення, нейромережа прямого розповсюдження (англ. *Feedforward neural network*) — це вид нейронної мережі, в якій сигнали поширюються в одному напрямку, починаючи від вхідного шару нейронів, через приховані шари до вихідного шару, і на вихідних нейронах отримується результат опрацювання сигналу [12]. В

мережах такого виду немає зворотніх зв’язків. Протилежним видом нейронних мереж зі зворотними зв’язками є рекурентні нейронні мережі. Прикладом нейронної мережі прямого поширення є персептрон Розенблатта, від якого і беруть свій початок нейромережі прямого розповсюдження [14]. В літературі часто термін перцептрон, багатошаровий перцептрон та нейромережа прямого поширення застосовуються як синоніми. Власне, між різними видами перцептронів спільне одне — вони усі є нейромережами з прямим поширенням сигналу, різняться, в основному, кількістю шарів, функцією активації та методом навчання. На рисунку 1.4 наведена модель персептрона.

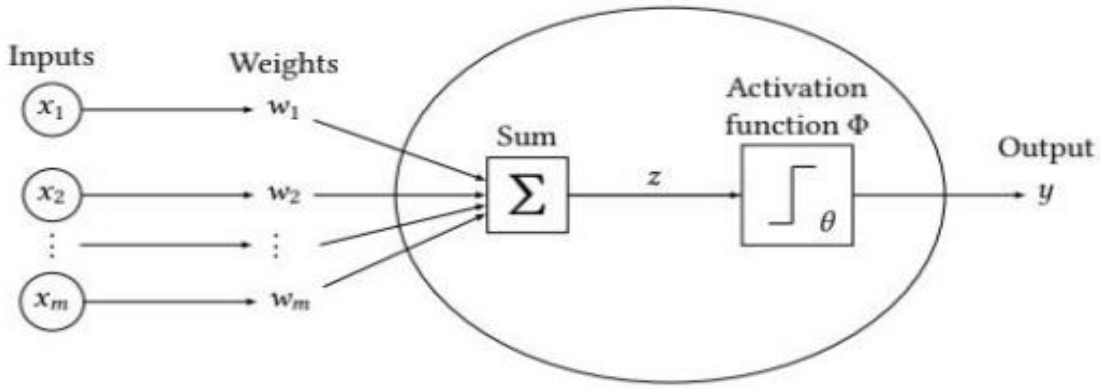

Рисунок 1.4 - Схема персептрона

1.11 RNN

Рекурентні нейронні мережі (RNN) - це клас нейронних мереж, які широко використовуються для моделювання послідовних даних, таких як природна мова. RNN- є логічним розвитком в послідовних нейронних мереж. На практиці RNN часто використовуються для послідовних входів. Особливо в мові, слова яких повинні бути оброблені крок за кроком, щоб відновити їх контекст. Також RNN виявилися дуже хорошими в задачах перекладу, оскільки вони можуть передбачити наступне слово за попередніми, більш ранніми словами. Щоб контролювати процес аналізу зразка з іншим, у RNN є приховані блоки для зберігання векторів стану. Ці вектори містять інформацію про

історію послідовності [6]. Таким чином, вихід *y* для RNN залежить від фактичного входу ⋀ *x* і попередніх входів (, ...,). Крім того, приховані стани представлені *t x1 xt−1* прихованою векторною послідовністю *h = (h1 , ..., ht*). Вихідні дані RNN складають послідовність *y* = (*y* , ...,), але не обов'язково представляють *1 yt* вихідні дані для загальної задачі. RNN також можуть використовуватися для прогнозування єдиної мети для вхідної послідовності. Прихований стан *ht* розраховується за формулою:

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

H означає приховану функцію шару, яка часто є гіперболічним тангенсом (tanh). Крім того, *W* позначає є вагову матрицю між шарами, а *b* представляє вектор зміщення. Вихід розраховується аналогічно результату багатошарового перцептрона шляхом множення ваги на результат прихованого стану і додавання зсуву. Тому прихований стан схожий на прихований шар.

Результат розраховується аналогічно:

$$y_t = H\left(W_{hy}h_t + b_y\right)$$

Вихід *y* і прихований стан пов'язаний з ваговою матрицею. Ця *t ht W hy* структура шарів порівнюється з одношаровим персептроном. RNN можна розглядати як складений багатошаровий персептрон з циклами, оскільки результат попереднього введення також є входом для поточного екземпляра.

Одним з недоліків стандартних RNN є проблема зникнення і вибуху градієнтів. Перша проблема змушує RNN відстежувати залежності в більш довгих вхідних послідовностях. Це пов'язано з двома причинами. По-перше, гіперболічний тангенс і сигмовидна функція, які часто використовуються в RNN для активації, насичуються дуже швидко, і тому їх градієнт стає ближчим до 0. По-друге, застосовуючи BPTT, градієнт експоненціально зменшується

шляхом множення його на повторювані вагові матриці. Це також змушує градієнт наближатися до нуля дуже швидко. Феномен вибуху градієнта приводить до значних коливань ваги мережі і збільшує час навчання, що може привести до відмови мережі.

Висновки до першого розділу

У даному розділі розглянуті та проаналізовані базові принципи обробки природної мови (Natural language processing) та сучасні методі використання машинного навчання та нейронних мереж для обробки текстів, представленими природними мовами. На основі проведеного аналізу та огляду сформована задачі та мета даної роботи.

Показано, що задача класифікації та обробки природних мов була сформована дуже давно, і за останній час набувають швидкого та стрімкого розвитку методи та методики, призначені для вирішення таких задач. Доведено, що автоматичне порівняння та класифікація текстів дає можливість додатково захистити інтелектуальну власність, здешевити процеси патентування, прискорити пошук необхідної інформації тощо.

Показані області використання технології NLP, яка на сьогодні широко використовується в голосових помічниках, автовідповідачах, оцифруванні мови, створення субтитрів, знаходження взаємозв’язків у текстах, виведення головної ідеї з тексту, аналіз дискурсу, прогнозування та підказки при друці, виправлення граматичних, синтаксичних, лексичних помилок тощо.

2 МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ОБРОБКИ ПРИРОДНИХ МОВ ТА (TEXT SUMMARIZATION)

2.1 Text summarization

Узагальнення тексту - text summarization - це короткий текстовий опис одного або декількох пунктів статті. Також може бути визначено як процес виділення найважливішої інформації з джерела (або джерел) для створення скороченій версії для конкретного користувача (або користувачів) і завдання (або завдань)[10].

Коли це робиться за допомогою комп'ютерних технологій, тобто автоматично, ми називаємо цей процес «автоматичним узагальненням тексту». Незважаючи на те, що традиційно була зосереджена на вводі тексту, інформація також може бути представлена і в мультимедійній формі, наприклад, зображення, відео або аудіо, а також онлайн-інформація або гіпертекст. Крім того, ми можемо говорити про узагальненні одного або декількох документів

Вихід моделі може проводитися в форматі якоїсь витримки (тобто коли виконується вибір «важливих» пропозицій в документі) або анотація / резюмування (коли узагальнення може служити заміною оригінальному документу).

Що стосується стилю вихідних даних, зазвичай проводиться чітке розходження між індикативними і інформативними узагальненнями. Перший використовується для визначення тем, які розглядаються в початковому тексті і можуть дати короткий уявлення, про що йде в ньому мова, а другий призначений для висвітлення тем початкового тексту.

2.2 Абстрактні та Екстрактивні методи узагальнення даних

На сьогодні для узагальнення тексту використовуються методи машинного навчання. Існує два основних метода узагальнення: ектсрактивні та абстрактні (рисунок 2.1).

Рисунок 2.1-Схема класифікації узагальнення

Техніка узагальнення екстрактивного тексту передбачає витягування ключових фраз та слів із вихідного документа та їх комбінування для створення узагальнення тексту. Витяг проводиться відповідно до визначеної метрики без внесення змін до текстів.

Основними методами, роботу яких можна охарактеризувати як екстрактивне узагальнення є:

- Query Based and Generic Summarization - під час узагальнення тексту на основі запитів оцінка речень документа, базується на підрахунку частоти;
- слова або фрази. Вищі бали отримують речення, що містять фрази запиту, а не одиничні слова, які запитуються. Потім речення з найвищими оцінками витягуються для підсумкового аналізу разом з їх структурою контексту. Частини тексту можна витягувати з різних розділів або підрозділів. Отриманий підсумок об'єднується в „екстракти“ узагальнення тексту. В алгоритмі вилучення речень, кожен

раз, коли речення вибирається для включення до узагальненого тексту, деякі заголовків у цьому контексті також вибрані;

- прихована модель Маркова - прихована модель Маркова (ПММ) - ще один метод вилучення речення з документа. На відміну від наївного Байєсовського підходу, прихована модель Маркова має менше припущень про незалежність. Зокрема, якщо ПММ цього не робить, припустимо, що ймовірність того, що речення *i* є у зведенні, та не залежить від того, чи є речення *i*-1 в узагальнення речення. Основна ідея полягає у використанні послідовної моделі для врахування місцевих залежностей між реченнями. У моделі ПММ були використані наступні функції:
    1) положення речення в документі;
    2) кількість термінів у реченні%
    3) правдоподібність термінів речення з урахуванням термінів документа.

Одне з обмежень статистичного підходу полягає в тому, що вони не враховують семантичне відношення серед речень. Тобто використовуючи цей метод можна зберегти інформацію при передачі, але граматична коректність сформованих речень буде, на жаль, не дуже гарною.

Узагальнення на основі абстракції передбачає перефразування та скорочення частин вихідного документа. Коли абстракція застосовується для узагальнення тексту в проблемах глибокого навчання, це може подолати граматичні невідповідності методу екстракції.

Алгоритми узагальнення абстрактного тексту створюють нові фрази та речення, які передають найбільш важливу (корисну) інформацію з оригінального тексту, як це робиться людиною. Це стає можливим завдяки новому типу нейронних мереж з використання великих корпусів текстів для навчання, а саме використанням RNN моделей та покращеним методам векторизації слів (word embeddings). По своїй суті такі мережі генерують новий

текст, базуючись на інформації старого тексту, проте вони не обмежені словником старого тексту, а використовують самий повний можливий словник векторного представлення слів.

Тому абстракція працює ефективніше, ніж екстракція. Однак алгоритми узагальнення тексту, необхідні для абстрагування, важче розробити, оскільки це потребує великої кількості підготовлених даних, великих моделей з багатьма параметрами, а для навчання таких моделей необхідні кластери чи високопотужні комп'ютерні системи.

2.3 FastText

FastText - це бібліотека для створення векторних представлень слів, розроблена дослідницькою групою Facebook AI. Цю бібліотеку можна порівняти з word2vec, але вона використовує символьні *n*-грами в якості вхідних даних, а не тільки саме слово. Це покращує здатність створювати вектори навіть для вигаданих або рідкісних слів. Крім того, цей метод підходить для мов, які мають багато перегинів і засновані на одних і тих же невеликих частинах слова, таких як фінська або турецька. FastText використовують CBOW або Skip-грами, які обговорювалися в попередньому розділі. У word2vec Skip-gram прогнозує вектор контекстних слів (вихідний), отримавши фактичне слово w, використовуючи закодовані вектори слів. Ці представлення не мають спільних рис між словами, оскільки вони можуть тільки вказати, чи є слово тим же самим чи ні. FastText, навпаки, розбиває свої вхідні слова на набори символів *n*-грам і передає їх в мережу. Символьні *n*-грами можуть бути розділені між різними словами. Наприклад, слово «дослідження» ділиться на "<до, дос, осл, осл, слі, лід, ідж, дже, жен, енн, ня>" якщо n дорівнює 3. Таким чином, «<до» і «ня>» містять спеціальні символи «<» і «>», що вказують початок і кінець слова. Це зроблено для того, щоб проілюструвати різницю між словами, які містяться у вигляді *n*-грами символів в іншому слові, наприклад, «слід» в «дослідженні».

Для отримання вектора слова для слова *w* в даному наборі документів *t* вводиться функція оцінки s(w ,). Вона обчислює суму символів *n*-грам, *t wc* , помножених на навколишнє слово *w* ∈ *{..., wt - 2, wt - 1, wt + 1, wt + 2, ...}* зі слова *w* . Кількість контекстних слів задається як довжина вікна виведення *t wt* Skip-грам. Skip-грам-версія fastText розраховує умовну ймовірність:

$$p(w_c|w_t) = \frac{e^{S(w_t,w_c)}}{\sum_{i=1}^{N} e^{s(w_t,w_c)}}$$

замінивши результат скалярного множення між *ucv* на *s*(,). Нехай *G - t T wt wc* словник *n*-грам, а *G ⊂ {1, ..., G}* - це масив символьних *n*-грам для слова *t wt* Тоді результат

$$s(w_t|w_c) = \sum_{g=G_t} z_{gC_c}^T$$

визначається для кожного контекстного слова *wc* . Замість унітарно-кодованого слова вектора набір символів *n*-грамів прогнозує контекстне слово у FastText. Усі вектори *z* де *g* ∈ є символьними вкладеннями *n*-грам слова . Їх *d Gt wt* набір будує ембедінг слова *v* . Через те, що загальний набір *n*-грам для *t* великого корпусу дуже великий, хешована версія символів *n*-грам використовується як вхідна інформація. Перевагою FastText є продуктивність. Час навчання представлень слів FastText короткий, і навіть можна навчити вектори на стандартному багатоядерному процесорі в реальні терміни. Більше того, точність FastText у завданнях NLP, таких як семантичний аналіз або класифікація за тегами, одна з найвищих серед найсучасніших методів.

2.4 GPT-2 (мовна модель)

Мовна модель рисунок 2.2- по суті, це модель машинного навчання, яка дивиться на кілька слів у реченні і пророкує наступне слово. Найбільш відома мовна модель - це клавіатура смартфона, яка під час набору тексту підказує вам продовження.

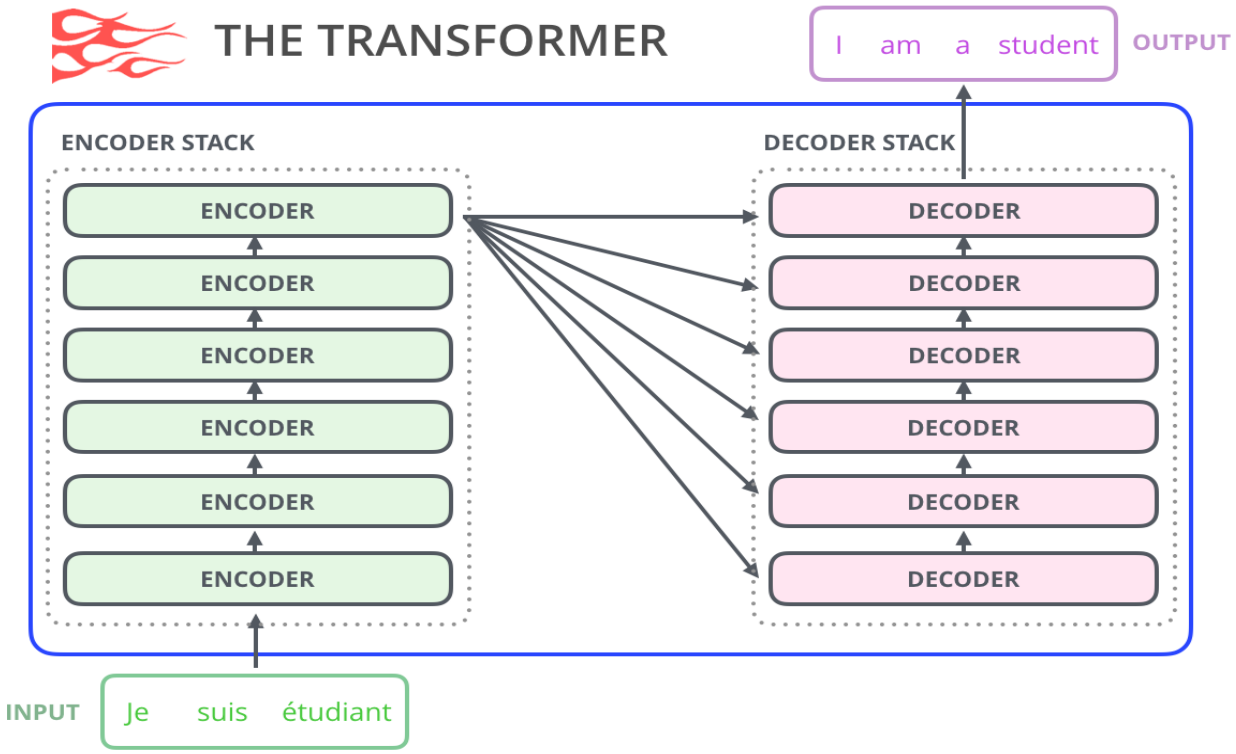

Рисунок 2.2 - Схема блока мовної моделі «трансформер»

У цьому сенсі можна сказати, що GPT-2 представляє собою алгоритм передбачення наступного слова клавіатурного додатка, але більш великоваговий і розумний, ніж той, що реалізований у простому мобільному телефоні. GPT-2 була навчена на великому наборі даних розміром 40 Гб (WebText), який OpenAI зібрали з мережі Інтернет в рамках свого дослідницького проекту. З точки зору обсягу зберігання даних, клавіатурний додаток, наприклад, SwiftKey, займає до 78 Мб, в той час як найменший варіант навченої GPT-2 використовує вже 500 Мб для зберігання всіх її параметрів, а найбільша модель GPT-2 - в 13 разів більше (так що вона може

займати до 6,5 Гб). Раніше дуже популярною була модель трансформера. Модель трансформера складалася з енкодера і декодера, кожен з яких представляє собою стек блоків Трансформера. Ця архітектура підходила для машинного перекладу - завдання, де архітектури енкодер-декодер показували хороші результати і в минулому.

Наступні за релізом дослідження показали, що можна відкинути декодер або енкодер і використовувати всього один стек блоків Трансформера, налаштовуючи ці блоки стільки, скільки взагалі можливо, передаючи їм дуже великі обсяги текстових даних для навчання і виконуючи величезні обсяги обчислень на них (деякі з цих моделей вимагають сотні тисяч доларів для навчання і навіть мільйони у випадку з AlphaStar).

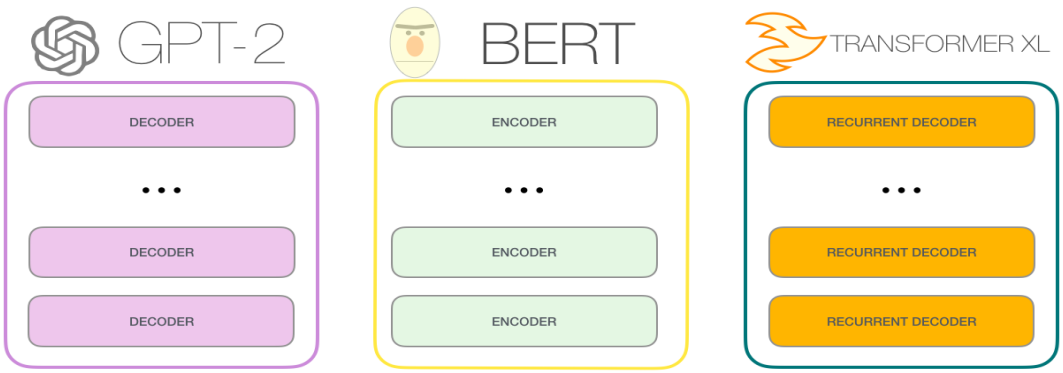

Рисунок 2.3 - Особливості основних блоків мовних нейронних моделей

Моодель GPT-2 побудована за допомогою блоків декодера Трансформера. BERT і, навпаки, використовує блоки енкодера. Проаналізуємо різницю двох підходів в наступному розділі. Але одна ключова відмінність полягає в тому, що GPT-2, як і всі традиційні мовні моделі, генерує на виході один токен за один раз.

Насправді відбувається наступне: після того, як був обчислений кожен токен, він додається до вхідної послідовність. І ця нова послідовність подається на вхід моделі на наступному кроці. Ця ідея називається

«авторегресії» (auto-regression) і саме вона зробила RNN мережі невиправдано ефективними.

GPT-2 і деякі нові моделі на кшталт TransformerXL і XLNet авторегрессівни за своєю природою. BERT немає. І це свого роду компроміс. Втративши авторегресії, BERT набув здатності включати контекст по обидва боки слова для отримання кращих результатів. XLNet повернув собі авторегресії, знайшовши при цьому альтернативний спосіб інкорпорувати контекст по обидва боки.

2.5 Мультиноміальна логістична регресія

При обробці природної мови класифікатори, що базуються на мультиноміальній логістичній регресії, зазвичай використовуються в якості альтернативи наївному Баєсовому класифікатору, оскільки вони не передбачають статистичну незалежність випадкових величин (які, зазвичай, називаються ознаками), які служать предикторами. Однак навчання в такій моделі повільніше, ніж для наївного байєсівського класифікатора, і, отже, може бути недоречним, враховуючи дуже велику кількість класів для навчання. Зокрема, навчання в наївному байєсівської класифікаторі - це простий підрахунок кількості одночасних появ об'єктів і класів, в той час як у мультиномальних логічних регресіях класифікаторах ваги, які зазвичай максимізуються з використанням максимальної апостеріорної оцінки, повинні навчатися з використанням ітераційної процедури. У статистиці поліноміальна логістична регресія - це метод класифікації, який узагальнює логістичну регресію для мультикласових задач, тобто з більш ніж двома можливими дискретними результатами. Тобто це модель, яка використовується для прогнозування ймовірностей різних можливих результатів категоріально розподіленої залежною змінною, враховуючи набір незалежних змінних, які можуть бути речовими, двійковими, категоріальним і т. д.

Поліноміальна логістична регресія має безліч інших імен, у тому числі багаточленна логістична регресія, мультикласова регресія, регресія softmax, mlogit, класифікатор максимальної ентропії (MaxEnt) і умовна модель максимальної ентропії.

2.6 Вибір інструментарію для вирішення поставленої задачі.

Для вирішення даної задачі була використана мова програмування Python. Python-багаторівнева мова програмування, яка містить безліч різноманітних способів застосування. Особистий статус йому надає багаторазове співтовариство розробників машинного навчання.

Завдяки активному розвитку для Python з'явилося безліч готових бібліотек машинного навчання. Ця мова - платформно незалежна, тому її можна адаптувати практично до будь-якої операційної системи. Ще одне перевагу Python пов'язано з його відкритістю - він побудований на базі технологій Open Source, тому розробники можуть отримати доступ до будь-якої мови.

Для роботи із текстовими даними необхідно проходити попередню обробку тестів. Для цього було обрано NLTK і spaCy.

Проаналізуємо основні характеристики цих бібліотек та визначимо їх можливості для вирішення поставлених завдань.

NLTK - це важлива бібліотека, яка підтримує такі завдання, як класифікація, стемінг, маркування, синтаксичний аналіз і семантичне міркування в Python. Це основний інструмент для обробки природної мови та машинного навчання. Сьогодні він служить освітньої основою для розробників Python, які тільки розпочинають вивчення NLP і машинного навчання.

Бібліотека була розроблена Стівеном Бердом і Едвардом Лопера в Пенсильванському університеті [8]. Вона зіграла ключову роль в проривних дослідженнях NLP. NLTK, поряд з іншими бібліотеками та інструментами

Python, тепер використовують в своїх навчальних програмах університети по всьому світу.

Бібліотека досить універсальна, однак її важко використовувати для обробки природної мови. NLTK може бути досить повільним і не відповідати вимогам, що швидко розвиваються для виробничого використання.

SpaCy відносно молода бібліотека, призначена для виробничого використання. Тому вона набагато доступніше інших NLP-бібліотек Python, таких як NLTK. Бібліотека spaCy пропонує найшвидший синтаксичний парсер, наявний сьогодні на ринку. Крім того, оскільки інструментарій написаний мовою Python, він також дуже швидкий і ефективний.

Але жоден інструмент не є досконалим. У порівнянні з бібліотеками, які розглянуто в роботі, spaCy підтримує найменша кількість мов (на сьогодні сім). Однак зростаюча популярність машинного навчання, NLP і spaCy як ключовий бібліотеки означає, що цей інструмент може незабаром почати підтримувати більше мов програмування.

Висновки до другого розділу

У даному розділі були розглянуті та проаналізовані актуальні методики з рішення проблем обробки натуральних мов (Natural language processing) та сучасні методі використання машинного навчання та нейронних мереж для обробки природної мови.

Проаналізовано проблеми векторного представлення природомовних об’єктів інформації, продовження (угадування) природомовної форми інформації з використанням машинного навчання. Використання векторних представлень слів у обробці природних мов дає велику перевагу перед більш простими методами і дозволяє знаходити додаткові, неочевидні взаємозв’язки між текстами. Проведено порівняння існуючих рішень, а також наведено оглядовий вступ в машинне навчання.

Обґрунтовано вибір інструментарію для вирішення поставлених задач.

3 РОЗРОБКА МОДЕЛІ НЕЙРОННИХ МЕРЕЖ ТА ПРОЕКТУВАННЯ АРХІТЕТУРИ (ПРОГРАМНОГО КОМПЛЕКСУ)

3.1 Пошук даних

Однією з головних проблем в рішенні задач обробки натуральних мов є складання необхідного набору даних, на яких буде навчатись нейронна мережа.

Для складання необхідного набору даних витрачається дуже багато часу. Тому складати власний корпус текстів не є доцільним в задачах дослідження. Для вирішення даної проблеми, було вирішено використати готовий набір даних, складений англійською мовою, оскільки саме англійською мовою існує найбільш повні та великі набори даних.

Тому для навчання та тестування нейронних мереж було використано набір даних з сайта Kaggle. Цей сайт є найбільш популярною платформою з поширення наборів даних для навчання нейронних моделей. Також на ньому проводять змагання та навчання з обраного напрямку машинного навчання. Цей набір даних містить близько 200 тис. Заголовків новин з 2012 по 2018 рік, отримані від HuffPost [3]. Модель, навчена на цьому наборі даних, може бути використана для ідентифікації тегів для відстежених статей новин або для визначення типу мови, що використовується в різних статтях новин. Кожен заголовок новин має відповідну категорію. Категорії та відповідна кількість статей, в даному датасеті 37 категорій. Описаний датасет в JSON форматі.

3.2 Вибір метрик для порівняння методів та моделей

Для оцінки якості навчання нейронної мережі доцільно мати декілька метрик що відображають якість навчання та дозволяють зробити кількісне порівняння.

Часта правильних класифікованих об’єктів(Accuracy) — ймовірність того, що клас буде передбачений правильно серед повної множини об’єктів і визначається наступним чином.

$$Accuracy = \frac{1}{2}\sum_{x \in S} \frac{tp_x + tn_x}{tp_x + fn_x + tn_x + fp_x}$$

де:

tpx – кількість об’єктів класу x, що віднесені алгоритмом до класу x

fpx – кількість об’єктів не класу x, що віднесені алгоритмом до класу x;

fnx – кількість об’єктів класу x, що віднесені алгоритмом не до класу x;

tnx – кількість об’єктів не класу x, що віднесені алгоритмом не до класу x;

S – множина усіх класів, до котрих відносяться об’єкти вибірки

Точність (Precision) показує, яку частку об'єктів, розпізнаних як об'єкти необхідного класу, було передбачено вірно і визначається наступним чином.

$$Precision = \frac{TP}{TP + FP}$$

Повнота (Recall) — це повнота показує, яку частку об'єктів, які реально належать до необхідного класу, було передбачено вірно.

$$Recall = \frac{TP}{TP + FN}$$

K-fold Cross-validation — *k*-кратна перехресна перевірка, рисунок 3.1. Метод формування навчальної та тестової множини для навчання аналітичної моделі в умовах недостатності вихідних даних або нерівномірного представлення класів.

Для успішного навчання аналітичної моделі необхідно, щоб класи були представлені в навчальній множині приблизно в однаковій пропорції. Однак якщо даних недостатньо або процедура семплінгу при формуванні навчальної множини була проведена невдало, один з класів може виявитися домінуючим. Це може викликати «диспропорцію» в процесі навчання, і домінуючий клас

буде розглядатися як найбільш імовірний. Метод перехресної перевірки дозволяє уникнути цього.

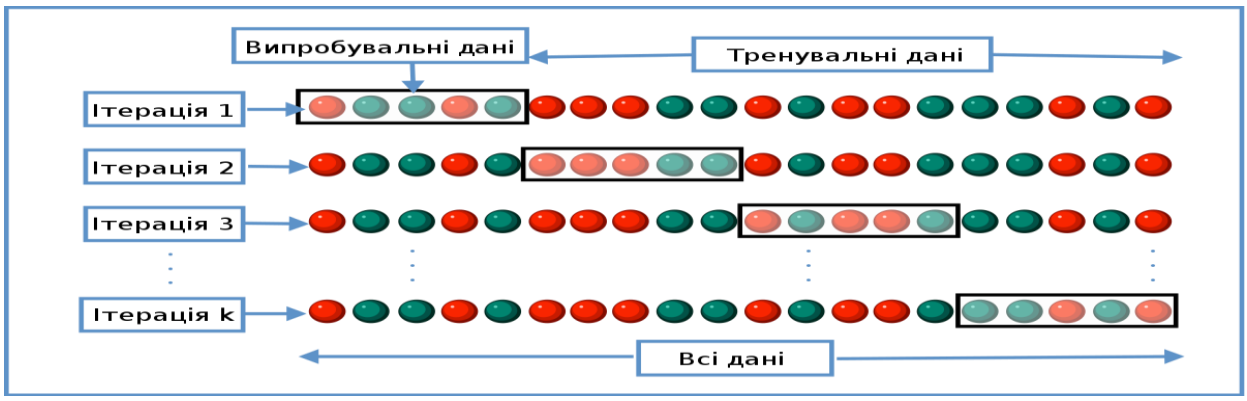

Рисунок 3.1-Схема k-кратного перехресного затверджування.

3.3 Порівняння методик

Для задачі порівняння текстів та визначення коефіцієнту подібності, необхідно мати математичне представлення слів. З цією метою використовується Word2Vec - методика векторного представлення слів[2].

Використовуючи Word2Vec представити схожість між текстами можна за допомогою таких наступних методів.

Подібність косинусів - це міра подібності двох ненульових векторів внутрішнього словарного простору, що вимірює косинус кута між ними.

Словарна відстань - це міра, яка характеризує відстань, яку повинні “проїхати” слова із одного документа. Чим вона низча, тим більша спорідненість.

Евклідова відстань - ця метрика дозволяє визначити, наскільки дві точки або два вектори віддалені один від одного. Використовує норму різниці між векторами.

Завдяки векторному представленню слів стало зручним використовувати текстові дані в машинному навчанні. Для знаходження семантичної схожості між словами та векторами слів, прийнято використовувати машинне навчання.

Методи машинного навчання, які проявили себе в розпізнаванні зображень, довели, що використання машинного навчання із вчителем, є ефективним способом вирішення задач, які потребують універсального підходу. Тому не дивно, що представлення слів у векторній формі стало поштовхом для створення машинних моделей завдання, які здатні вирішувати задачі обробки натуральних мов.

Одної із таких моделей для вирішення загальних задач з класифікації та генерації тексту є GPT-2. GPT-2 - це модель машинного навчання, яка спеціалізується на текстовій генерації текста. Основним блоком в даній моделі є блок декодера, який був взятий із іншої моделі машинного навчання “Трансформер”. Структура GTP-2 дозволяє захоплювати не одне слово, а весь контекст слова в реченні.

Використовуючи попередню навчену модель та набір фейкових новин. Було створення порівняння методів. Результати проведеного порівняння наведено у таблиці 3.1.

Таблиця 3.1-Таблиця порівняння методів

Комбінація методів	Результативність на навчальному наборі	Результативність на тестовому наборі
GPT-2 + подібність косинусів	84.1%	80.3%
GPT-2 + словарна відстань	81.1%	79.1%
GPT-2 + евклідова відстань	81.3%	79.6%
Модель машинного навчання без попереднього навчання + подібність косинусів	40.3%	36.7%

Використовуючи дані, отримані емпіричним шляхом, було визначено, що одним із найбільш ефективних методів порівняння тексту є комбінація методів машинного навчання з попереднім навчанням та косинусної подібності.

3.3 Використання метода узагальнення даних

Із загальним розширенням вхідних даних, розмірів докуммента. Стає гостра проблема в коректному порівнянні данних, а саме із збільшенням розміру документа стає ваще визначити принадлежність документа до певної конкретної теми, що призводить до логічного вирішення проблеми. Потрібно зменшити кількість вхідних даних не втративши разом із цим ціної інформації. Зберегти так звані опорні словарні вектори на базі яких і буде проводитись порівняння. Таким чинимо вирішеться проблема в розмірності даних, та зайвого інформаційного шуму.

Для навчання мережі використовується загальнодоступний набір даних. Цей набір даних складається з оглядів шуканих продуктів від Amazon. Дані охоплюють період понад 10 років, включаючи всі ~ 500 000 оглядів до жовтня 2012 р. Ці огляди включають інформацію про продукт та користувачів, рейтинги, огляд у простому тексті та підсумок. Він також включає огляди з усіх інших категорій Amazon.

Було взято вибірку із 100 000 рядків, щоб скоротити час навчання нашої моделі.

```
data=pd.read_csv("../input/amazon-fine-food-reviews/Reviews.csv",nrows=100000)
```

Виконання основних кроків попередньої обробки дуже важливо перед тим, як перейти до частини побудови моделі. Використання брудних та неочищених текстових даних є потенційно згубним кроком. Отже, на цьому кроці було викинуто з тексту всі небажані символи.

Створимо словник скорочень, який буде складатись із таких слів.

```
contraction_mapping = {"ain't": "is not", "aren't": "are not","can't": "cannot", "'cause": "because", "could've": "could have", "couldn't": "could not",

"didn't": "did not", "doesn't": "does not", "don't": "do not", "hadn't": "had not", "hasn't": "has not", "haven't": "have not",

"he'd": "he would","he'll": "he will", "he's": "he is", "how'd": "how did", "how'd'y": "how do you", "how'll": "how will", "how's": "how is",

"I'd": "I would", "I'd've": "I would have", "I'll": "I will", "I'll've": "I will have","I'm": "I am", "I've": "I have", "i'd": "i would",

"i'd've": "i would have", "i'll": "i will",  "i'll've": "i will have","i'm": "i am", "i've": "i have", "isn't": "is not", "it'd": "it would",

"it'd've": "it would have", "it'll": "it will", "it'll've": "it will have","it's": "it is", "let's": "let us", "ma'am": "madam",

"mayn't": "may not", "might've": "might have","mightn't": "might not","mightn't've": "might not have", "must've": "must have",

"mustn't": "must not", "mustn't've": "must not have", "needn't": "need not", "needn't've": "need not have","o'clock": "of the clock",

"oughtn't": "ought not", "oughtn't've": "ought not have", "shan't": "shall not", "sha'n't": "shall not", "shan't've": "shall not have",

"she'd": "she would", "she'd've": "she would have", "she'll": "she will", "she'll've": "she will have", "she's": "she is",

"should've": "should have", "shouldn't": "should not", "shouldn't've": "should not have", "so've": "so have","so's": "so as",

"this's": "this is","that'd": "that would", "that'd've": "that would have", "that's": "that is", "there'd": "there would",

"there'd've": "there would have", "there's": "there is", "here's": "here is","they'd": "they would", "they'd've": "they would have",

"they'll": "they will", "they'll've": "they will have", "they're": "they are", "they've": "they have", "to've": "to have",
```

"wasn't": "was not", "we'd": "we would", "we'd've": "we would have", "we'll": "we will", "we'll've": "we will have", "we're": "we are",

"we've": "we have", "weren't": "were not", "what'll": "what will", "what'll've": "what will have", "what're": "what are",

"what's": "what is", "what've": "what have", "when's": "when is", "when've": "when have", "where'd": "where did", "where's": "where is",

"where've": "where have", "who'll": "who will", "who'll've": "who will have", "who's": "who is", "who've": "who have",

"why's": "why is", "why've": "why have", "will've": "will have", "won't": "will not", "won't've": "will not have",

"would've": "would have", "wouldn't": "would not", "wouldn't've": "would not have", "y'all": "you all",

"y'all'd": "you all would","y'all'd've": "you all would have","y'all're": "you all are","y'all've": "you all have",

"you'd": "you would", "you'd've": "you would have", "you'll": "you will", "you'll've": "you will have",

"you're": "you are", "you've": "you have"}

Потрібно визначити дві різні функції попередньої обробки оглядів та генерації резюме, оскільки етапи попередньої обробки тексту та резюме дещо відрізняються.

Очищення тексту

Відображення перших 10 рядків набору даних дадуть уявлення про етапи попередньої обробки тексту:

data['Text'][:10]

```
0    I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product look
s more like a stew than a processed meat and it smells better. My Labr...
1           Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this wa
s an error or if the vendor intended to represent the product as "Jumbo".
2    This is a confection that has been around a few centuries.  It is a light, pillowy citrus gelatin with nuts - in this case Fil
berts. And it is cut into tiny squares and then liberally coated with ...
3    If you are looking for the secret ingredient in Robitussin I believe I have found it.  I got this in addition to the Root Beer
Extract I ordered (which was good) and made some cherry soda.  The fl...
4                                                    Great taffy at a great price.  There was a wide assortment of yummy
taffy.  Delivery was very quick.  If your a taffy lover, this is a deal.
5    I got a wild hair for taffy and ordered this five pound bag. The taffy was all very enjoyable with many flavors: watermelon, r
oot beer, melon, peppermint, grape, etc. My only complaint is there wa...
6    This saltwater taffy had great flavors and was very soft and chewy.  Each candy was individually wrapped well.  None of the ca
ndies were stuck together, which did happen in the expensive version, ...
7                                                   This taffy is so good.  It is very soft and chewy.  The flavors are
amazing.  I would definitely recommend you buying it.  Very satisfying!!
8                                                 Right now I'm mostly just sprouting this so my cats can ea
t the grass. They love it. I rotate it around with Wheatgrass and Rye too
9                                                  This is a very healthy dog food. Good for their digestion. Also
good for small puppies. My dog eats her required amount at every feeding.
Name: Text, dtype: object
```

Необхідно виконати наведені нижче завдання попередньої обробки даних:

Перетвори всі в малу літеру

Видалити теги HTML

Видалити будь-який текст усередині дужок ()

Видалити розділові знаки та спеціальні символи

Видалити стоп-слова

Видалити короткі слова

```
stop_words =
set(stopwords.words('english'))
```

```
def text_cleaner(text):
    newString = text.lower()
    newString = BeautifulSoup(newString, "lxml").text
    newString = re.sub(r'\([^)]*\)', '', newString)
    newString = re.sub('"','', newString)
    newString = ' '.join([contraction_mapping[t] if t in contraction_mapping else t for t
in newString.split(" ")])
    newString = re.sub(r"'s\b","",newString)
    newString = re.sub("[^a-zA-Z]", " ", newString)
    tokens = [w for w in newString.split() if not w in stop_words]
```

```
    long_words=[]
    for i in tokens:
        if len(i)>=3:                #removing short word
            long_words.append(i)
    return (" ".join(long_words)).strip()
cleaned_text = []
for t in data['Text']:
    cleaned_text.append(text_cleaner(t))
```

Повторимо такі ж операції з вихідними даними.

Надалі необхідно провести аналіз розмірів повідомлення

Проаналізуємо тривалість оглядів та резюме, щоб отримати загальне уявлення про розподіл довжини тексту, рисунок 3.2. Це допоможе зафіксувати максимальну довжину послідовності:

```
import matplotlib.pyplot as plt
text_word_count = []
summary_word_count = []
# populate the lists with sentence lengths
for i in data['cleaned_text']:
      text_word_count.append(len(i.split()))
for i in data['cleaned_summary']:
      summary_word_count.append(len(i.split()))
length_df = pd.DataFrame({'text':text_word_count, 'summary':summary_word_count})
length_df.hist(bins = 30)
plt.show()
```

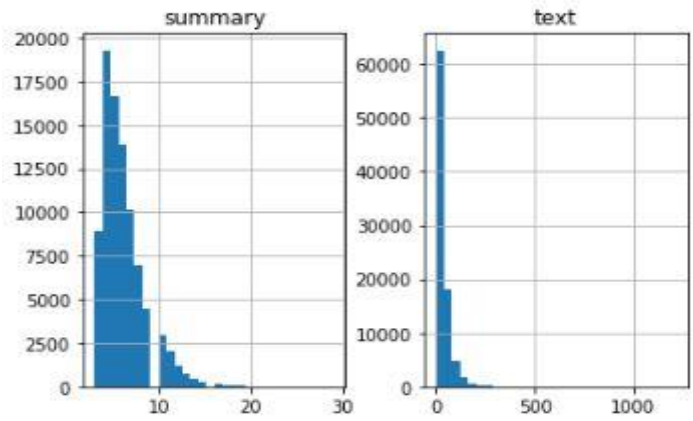

Рисунок 3.2 - Розподілення по довжині повідомлення

Встановлено максимальну тривалість оглядів до 80, оскільки це дасть більшість тривалості огляду. Аналогічним чином встановлено максимальну довжину підсумку 10:

```
max_len_text=80
                max_len_summary=10
```

Розділено набір даних на навчальний та тестовий набір.Використаємо 90% набору даних як навчальні дані та оцінимо ефективність на решти 10% :

```
from sklearn.model_selection import train_test_split
x_tr,x_val,y_tr,y_val=train_test_split(data['cleaned_text'],data['cleaned_summary'],test_size=0.1,random_
```

Побудова моделі.

Модель- трискладовий LSTM для кодера:

```
from keras import
backend as K
                K.clear_session()
                latent_dim = 500
```

```
# Encoder
encoder_inputs = Input(shape=(max_len_text,))
enc_emb = Embedding(x_voc_size,
latent_dim,trainable=True)(encoder_inputs)

#LSTM 1
encoder_lstm1 =
LSTM(latent_dim,return_sequences=True,return_state=True)
encoder_output1, state_h1, state_c1 = encoder_lstm1(enc_emb)

#LSTM 2
encoder_lstm2 =
LSTM(latent_dim,return_sequences=True,return_state=True)
encoder_output2, state_h2, state_c2 = encoder_lstm2(encoder_output1)

#LSTM 3
encoder_lstm3=LSTM(latent_dim, return_state=True,
return_sequences=True)
encoder_outputs, state_h, state_c= encoder_lstm3(encoder_output2)

# Set up the decoder.
decoder_inputs = Input(shape=(None,))
dec_emb_layer = Embedding(y_voc_size, latent_dim,trainable=True)
dec_emb = dec_emb_layer(decoder_inputs)

#LSTM using encoder_states as initial state
decoder_lstm = LSTM(latent_dim, return_sequences=True,
return_state=True)
decoder_outputs,decoder_fwd_state, decoder_back_state =
decoder_lstm(dec_emb,initial_state=[state_h, state_c])

#Attention Layer
Attention layer attn_layer = AttentionLayer(name='attention_layer')
```

```
attn_out, attn_states = attn_layer([encoder_outputs, decoder_outputs])

# Concat attention output and decoder LSTM output
decoder_concat_input = Concatenate(axis=-1,
name='concat_layer')([decoder_outputs, attn_out])

#Dense layer
decoder_dense = TimeDistributed(Dense(y_voc_size, activation='softmax'))
decoder_outputs = decoder_dense(decoder_concat_input)

# Define the model
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
model.summary()
```

Представлення моделі в вигляді графа зв’язаних шарів.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 80)           0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 80, 500)      25785500    input_1[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 80, 500), (N 2002000     embedding[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, None)         0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 80, 500), (N 2002000     lstm[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 500)    7048000     input_2[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 80, 500), (N 2002000     lstm_1[0][0]
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 500),  2002000     embedding_1[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
__________________________________________________________________________________________________
attention_layer (AttentionLayer [(None, None, 500),  500500      lstm_2[0][0]
                                                                 lstm_3[0][0]
__________________________________________________________________________________________________
concat_layer (Concatenate)      (None, None, 1000)   0           lstm_3[0][0]
                                                                 attention_layer[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 14096)  14110096    concat_layer[0][0]
==================================================================================================
Total params: 55,452,096
Trainable params: 55,452,096
Non-trainable params: 0
```

Навчання даної моделі використовуючи задані параметри.

```
history=model.fit([x_tr,y_tr[:,:-1]], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)[:,1:] ,epochs=50,callbacks=[es],batch_size=512, validation_data=([x_val,y_val[:,:-1]], y_val.reshape(y_val.shape[0],y_val.shape[1], 1)[:,1:]))
```

1. Побудова діагностичних діаграм рисунок 3.3 , щоб зрозуміти поведінку моделі з часом:

```
from matplotlib import
pyplot
                    pyplot.plot(history.history['loss'], label='train')
                    pyplot.plot(history.history['val_loss'],
                    label='test')
                    pyplot.legend() pyplot.show()
```

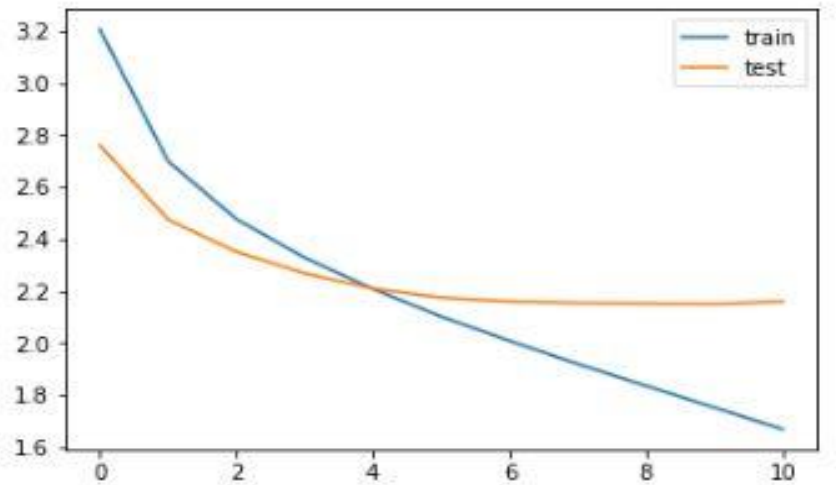

Рисунок 3.3 - Діаграма навчання машинної моделі

Висновок про незначне збільшення втрат від перевірки після десятої епохи . Отже доцільним буде припинити навчання моделі після цієї єпохи.

Далі, побудова словника для перетворення індексу в слово для цільової та вихідної лексики:

```
# encoder
inference
                encoder_model = Model(inputs=encoder_inputs,outputs=[encoder_outputs,
                state_h, state_c])

                # decoder inference
                # Below tensors will hold the states of the previous time step
                decoder_state_input_h = Input(shape=(latent_dim,))
                decoder_state_input_c = Input(shape=(latent_dim,))
                decoder_hidden_state_input = Input(shape=(max_len_text,latent_dim))

                # Get the embeddings of the decoder sequence
                dec_emb2= dec_emb_layer(decoder_inputs)

                # To predict the next word in the sequence, set the initial states to the states from
                the previous time step
                decoder_outputs2, state_h2, state_c2 = decoder_lstm(dec_emb2,
                initial_state=[decoder_state_input_h, decoder_state_input_c])

                #attention inference
                attn_out_inf, attn_states_inf = attn_layer([decoder_hidden_state_input,
                decoder_outputs2])
                decoder_inf_concat = Concatenate(axis=-1, name='concat')([decoder_outputs2,
                attn_out_inf])

                # A dense softmax layer to generate prob dist. over the target vocabulary
                decoder_outputs2 = decoder_dense(decoder_inf_concat)

                # Final decoder model
                decoder_model = Model(
```

```
[decoder_inputs] + [decoder_hidden_state_input,decoder_state_input_h,
decoder_state_input_c],
[decoder_outputs2] + [state_h2, state_c2])
```

Визначено функцію, нижче якої є реалізація процесу виведення:

```
def decode_sequence(input_seq):
# Encode the input as state vectors.
e_out, e_h, e_c = encoder_model.predict(input_seq)

# Generate empty target sequence of length 1.
target_seq = np.zeros((1,1))

# Chose the 'start' word as the first word of the target sequence
target_seq[0, 0] = target_word_index['start']

stop_condition = False
decoded_sentence = ''
while not stop_condition:
output_tokens, h, c = decoder_model.predict([target_seq] + [e_out, e_h, e_c])

# Sample a token
sampled_token_index = np.argmax(output_tokens[0, -1, :])
sampled_token = reverse_target_word_index[sampled_token_index]

if(sampled_token!='end'):
decoded_sentence += ' '+sampled_token

# Exit condition: either hit max length or find stop word.
if (sampled_token == 'end' or len(decoded_sentence.split()) >= (max_len_summary-
1)):
stop_condition = True
```

```
# Update the target sequence (of length 1).
target_seq = np.zeros((1,1))
target_seq[0, 0] = sampled_token_index

# Update internal states
e_h, e_c = h, c

return decoded_sentence
```

Функції для перетворення цілочисельної послідовності у послідовність слів для підсумку, а також оглядів:

```
def
seq2summary(input_seq):
                    newString=''
                    for i in input_seq:
                if((i!=0 and i!=target_word_index['start']) and
                i!=target_word_index['end']):
                        newString=newString+reverse_target_word_index[i]+' '
                    return newString

                def seq2text(input_seq):
                    newString=''
                    for i in input_seq:
                if(i!=0):
                        newString=newString+reverse_source_word_index[i]+' '
                    return newString



            print("Review:",seq2text(x_val[i]))
```

```
print("Original summary:",seq2summary(y_val[i]))
print("Predicted
summary:",decode_sequence(x_val[i].reshape(1,max_len_text)))
print("\n")
```

Результат роботи моделі на вийнятих декількох прикладів, які згенеровані моделлю.

```
Review: used eating flaxseed brownie hodgson mill brownies super easy make taste great since like dark chocolate usually add littl
e cocoa
Original summary: delicious brownie
Predicted summary:  best brownie mix
```

```
Review: favorite coffee keurig coffeemaker convenient get amazon cheaper running around stores trying find lowest price
Original summary: great coffee
Predicted summary:  great coffee
```

```
Review: mallomars pure chocolate cookies delicious tasty chocolate inside equally tasty cream filling inside pour ice cold glass mi
lk sit back try eat whole box one sitting brian fairbanks
Original summary: delicious
Predicted summary:  best chocolate have ever tasted
```

```
Review: organic usually prefer whatever blech cannot stand taste ended giving away going try another bag mention calories either be
ars calories take haribo please
Original summary: taste terrible
Predicted summary:  not that great
```

```
Review: package six boxes forty eight bags per box listed area large tea bags suitable making gallon time tea fact small single use
bags box web page says family size bags nothing family sized single use bags bad advertisement buy read misleading ads carefully ho
pe company business
Original summary: misleading advertisement
Predicted summary:  not as advertised
```

```
Review: red wine tart unpleasant way comes cans two servings per since carbonated either drink whole extended period save hope flat
share drink fairly quickly like normal soda get lot caffeine sugar pretty short time drinks like come smaller cans good perk right
point give jitters like drinks tend drank full two servings make heart anything drink several cups coffee day occasionally drink en
ergy drinks like well despite caffeine intake caffeinated soda like diet coke still keep night notably drink keep
Original summary: not bad has some ups and downs
Predicted summary:  not as good as it is
```

3.4 Класифікація тексту

Існує велика кількість алгоритмів машинного навчання задачою яких є класифікація тексту. Класичним являється методом є наївний Баєсів класифікатор. На попередньо описаному набору даних проведено навчання, та тестування Баєсівого класифікатора.

Для цього необхідно попередньо обробити текст як і з LSTM моделлю.

```
stop_words =
set(stopwords.words('english'))
                    def text_cleaner(text):
                      newString = text.lower()
                      newString = BeautifulSoup(newString, "lxml").text
                      newString = re.sub(r'\([^)]*\)', '', newString)
                      newString = re.sub('"','', newString)
                      newString = ' '.join([contraction_mapping[t] if t in
                    contraction_mapping else t for t in newString.split(" ")])
                      newString = re.sub(r"'s\b","",newString)
                      newString = re.sub("[^a-zA-Z]", " ", newString)
                      tokens = [w for w in newString.split() if not w in stop_words]
                      long_words=[]
                      for i in tokens:
                        if len(i)>=3:                #removing short word
                          long_words.append(i)
                      return (" ".join(long_words)).strip()

                    cleaned_text = []
                    for t in data['Text']:
                      cleaned_text.append(text_cleaner(t))
```

Далі за описаною в другому розділі методу створимо Баєсевський класифікатор. Для цього було використано бібліотеку sklearn, в якій уже є описаний вище Баєсівська модель. Проведемо навчання, рисунок 3.4 - схема роботи та навчання Баєсівського алгоритма

```
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB

dataset =dataset.news()
model = GaussianNB()
model.fit(dataset.data, dataset.target)
```

Для отримання та обробки точності роботи алгоритма на заданому наборі даних.

```
print("The accuracy is {}".format(accuracy_score(test_data.target,
predicted_categories)))
>The accuracy is 0.7638980350504514
```

Як результат роботи класичного метода класифікації тексту маємо точність на заданому наборі даних в 76%.

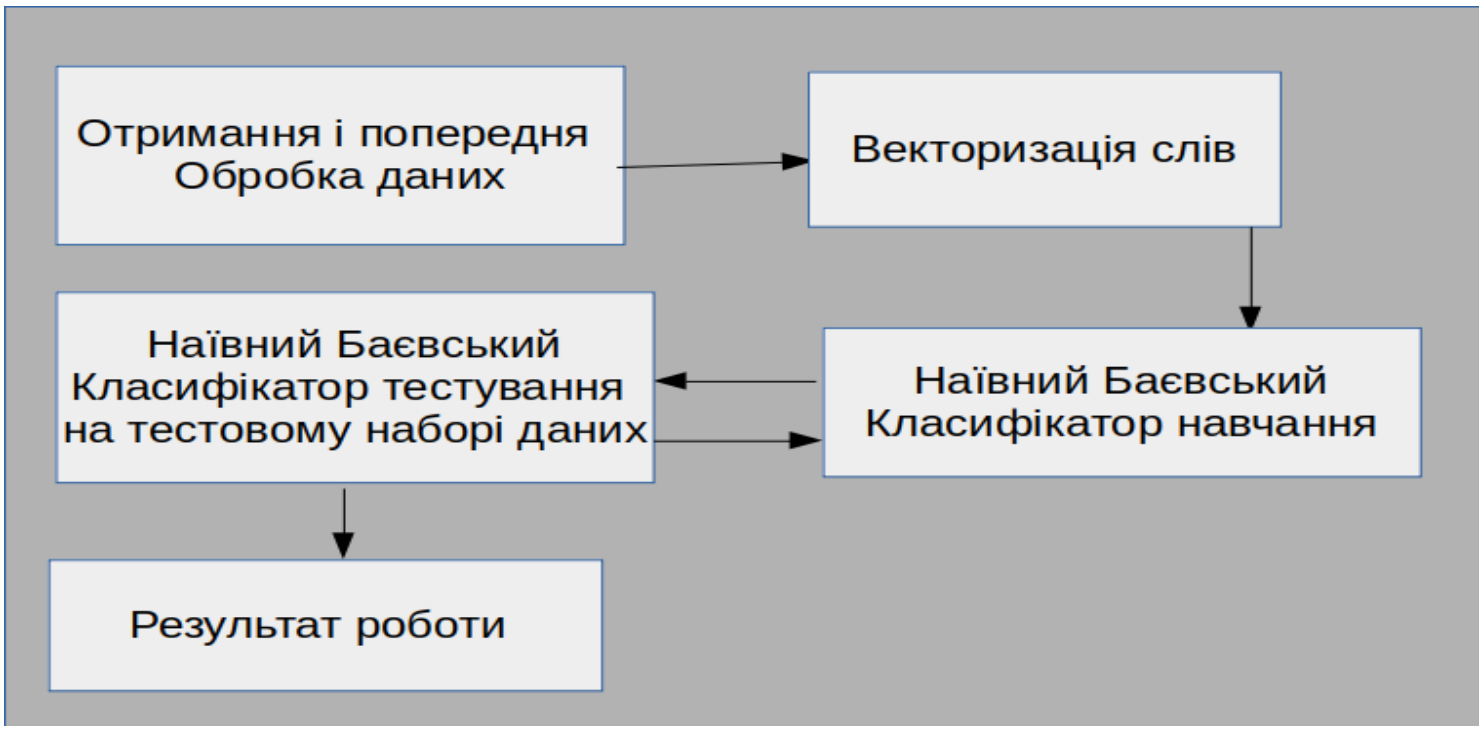

Рисунок 3.4 - Схема роботи та навчання Баєсівського алгоритма

3.5 Комбінації методів узагальнення даних і GPT-2

Класичні методи класифікації тексту являють собою класифікацію за векторами, які вони формують. Після процесу навчання на розмічених даних,

в яких є вхідний текст та його класифікаційні позначці методом машинного навчання з учителем досягається відповідність деякого абстрактного семантичного значення в відповідність до заданої тематики. По своїй суті на великому наборі даних, при малому вхідному розміру тексту відбувається навчання на векторному представленні даних.

На рисунку 3.5-представлення схема роботи метода GPT-2.

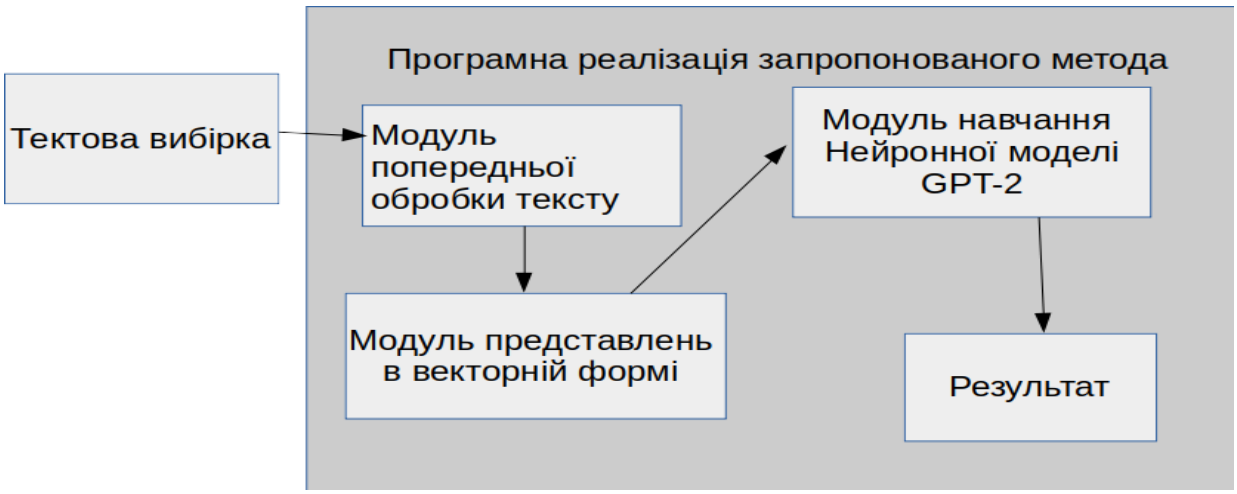

Рисунок 3.5- Програмна реалізація GPT-2 навчання

У таблиці наведено найкращі результати, виявлені для кожного LSTM набору гіперпараметрів . На рисунку 3.5 та таблиці 3.2 показані процеси навчання залежно від епох втрат для LSTM, розділені для читабельності. Ці узагальнені експерименти для всіх даних забезпечують ключову точку вибору найкращої моделі для вирішення задач обробки тексту.

Таблиця 3.2- процеси навчання залежно від епох втрат для LSTM

	Best Loss	Epoch
LSTM(128)	0.53	71
LSTM(128,128)	0.53	80
LSTM(128,128,12)	0.52	93

Тренувальні процеси найкращих моделей з точки зору втрат, відокремлені для читабельності. Результати наводяться для порівняльного аналізу для всіх наборів даних.

На рисунку 3.6 представлено порівняння LSTM моделей з різною кількістю шарів.

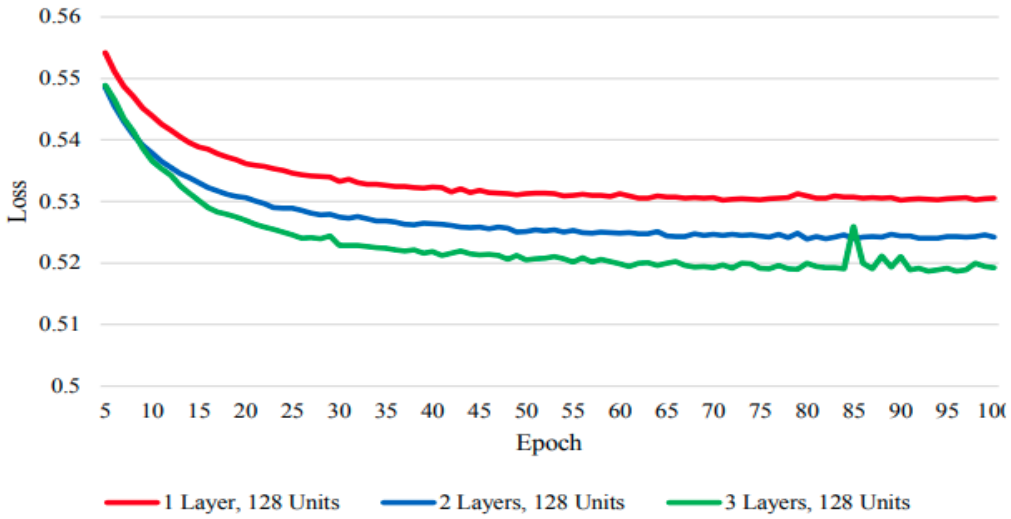

Рисунок 3.6 - Порівняння LSTM моделей з різною кількістю шарів

У таблиці 3.3 наведено результати дослідження, в комбінації методів LSTM та GPT-2 різними параметрами. Де параметри мінімальна кількість слів в одному реченні-означає кількість слів в реченні при якій узагальнююча мераже LSTM бере їх для узагальнення. Тобто менша кількість слів в реченні не додає це речення до узагальнення.

Таблиця 3.3 - Результати дослідження, в комбінації методів LSTM та GPT-2 різними параметрами

Розмірність LSTM-моделі	Мінімальна кількість слів в одному реченні	Кількість шарів Decoder в GPT-2	Розмір попередньо навченої GPT-2 моделі	Результат на тестовому наборі даних	Результат на навчальному наборі даних
LSTM 1layer	4	10	117M param	61.1%	64.1%
LSTM 1layer	4	30	345M param	63.0%	65.1%
LSTM 1layer	4	50	762M param	65.6%	67.3%
LSTM 2layer	7	10	117M param	60.1%	64.1%
LSTM 2layer	7	30	345M param	62.0%	65.1%
LSTM 2layer	7	50	762M param	63.6%	67.3%
LSTM 3layer	10	10	117M param	61.1%	64.1%
LSTM 3layer	10	30	345M param	63.0%	65.1%
LSTM 3layer	10	50	762M param	66.6%	68.3%

3.6 Порівняння роботи методик класифікації без використання узагальнення тексту та із узагальненням.

За отриманими даними в минулому розділі можна зробити висновок, що комбінація методів узагальнення, а потім класифікації даних не дає ніякого виграшу в порівнянні з класичним наївним Баєвським класифікатором. Але як було виявлено на першому датасеті, GPT-2 добре справляється з класифікацією даних. Для перевірки та виявлення ключового недоліка порівняємо роботу

GPT-2 моделі з перед навчанням та комбінації методів LSTM та GPT-2. Для цього оберемо найкращий результат із попередньої таблиці 3.3 та повторимо навчання. За отриманими результатами складемо таблицю 3.4.

Таблиця 3.4- порівняння роботи GPT-2 моделі із перед навчанням, та комбінації методів LSTM та GPT-2

	Результат на тестовому наборі даних	Результат на навчальному наборі даних
GPT-2 + LSTM	67.1%	68.4%
GPT-2	82.1%	84.4%

За отриманими даними можна зробити висновок, що запропонована схема з використанням узагальнення текстових даних не є ефективною. Це можна пояснити тим, що дані, проходячи через узагальнюючу мережу втрачають велику кількість інформації про свій контекст, і це значно погіршує результати. Тому класифікація значно гірша. На рисунку 3.7 показана схема програмної реалізації запропонованої комбінації методів LSTM та GPT-2.

Для вирішення цієї проблеми необхідно або змінити набір даних на яких тренується модель, що в цілому дозволить вирішити проблеми із браком контекста. Проте це не є доцільним, оскільки для зібрання необхідного набору даних потрібно багато часу, а також тоді даний методо з комбінації GPT-2 та LSTM буде хорошо працювати тільки на цьому наборі даних. На рисунку 3.7 — схематично зображено робота GPT-2 з даними після узагальнення.

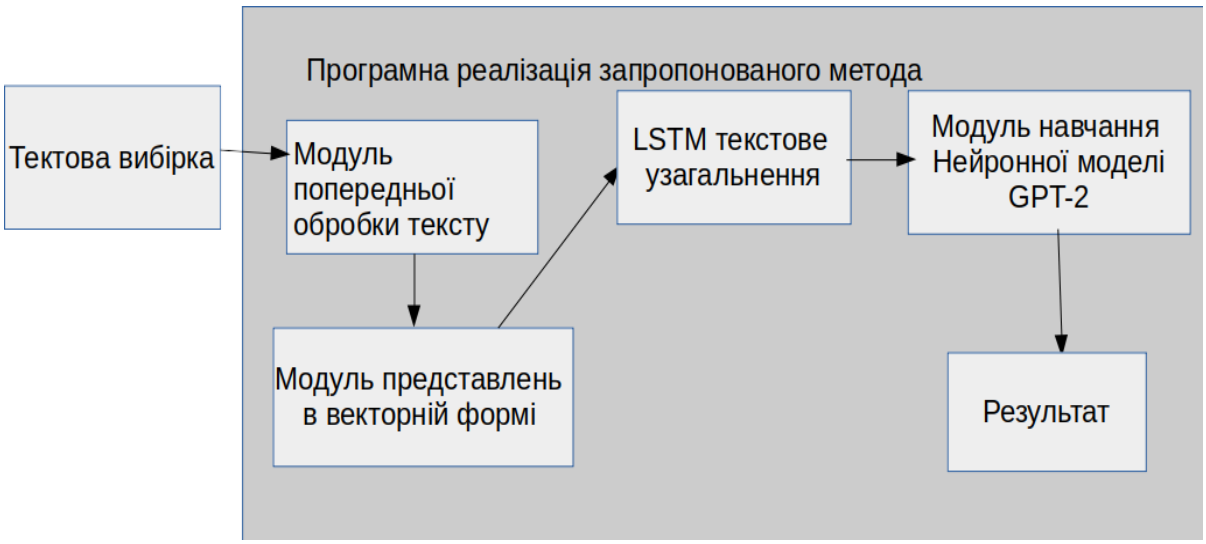

Рисунок 3.7 - Схема програмної реалізації GPT-2 + LSTM

Використання метода узагальнення даних має на меті зменшити навчальну вибірку та розширити сприйняття контексту із локально (речення, абзацу) до більш широкого (абзацу, розділу). Таким чином, включаючи в стиснутий текст виділений контекст за розділом або абзацом, залежно від заданих параметрів, отримаємо структуру моделі, представлену на рисунку 3.8.

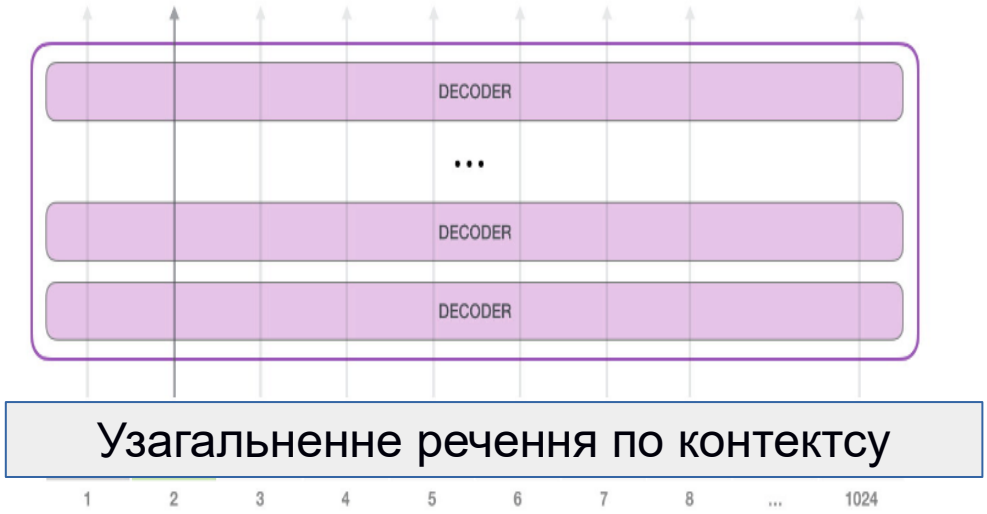

Рисунок 3.8- Робота GPT-2 та вигляд даних, які потрапляють на вхід моделі

Найбільш логічним розвитком такого методу є розширення вхідних даних за допомогою вихідних даних із LSTM моделі. Таким чином, на вхід LSTM моделі в залежності від обраних параметрів «мінімальна кількість слів», буде потрапляти текст, який обрамляє речення і є абзацом, або декілька абзаців, які обрамляють конкретний абзац. Тобто стане можливим подавати на вхід GPT-2 не весь текст документа, а лише необхідний текст та узагальнений вектор текстових даних. Таким чином, можна не просто зберігати контекст, а й розширяти його.

На рисунку 3.9 показано схему програмної реалізації даного метода, на вхід GPT-2 моделі потрапляють як вектор слів, так і узагальнений текст повного документа в формі вектора. Після додаткового навчання нової моделі. Складено порівняльну таблицю (таблиця 5) класифікації новин, за допомогою модифікованого метода та звичайного GPT-2 метода.

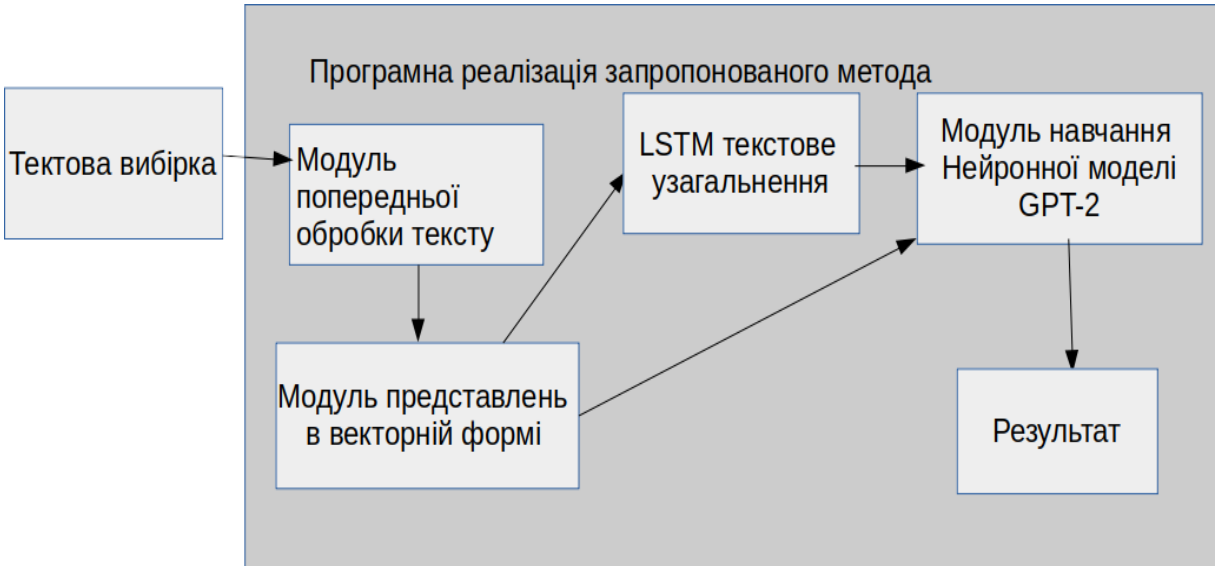

Рисунок 3.9 - Схема покращеної програмної реалізації GPT-2 + LSTM

Таблиця 3.5- порівняльну таблицю класифікації новин

	Результат на тестовому наборі даних	Результат на навчальному наборі даних
GPT-2 + LSTM модифікований	84.2%	86.4%
GPT-2	82.1%	84.4%
GPT-2 + LSTM	67.1%	68.4%

Як можна побачити з таблиці 3.5, було отримано покращення в правильності класифікації новин відносно звичайного GPT-2 метода. Та що не менш важливо, бачимо значну перевагу модифікованого метода в порівнянні з не модифікованої комбінації методів GPT-2 та LSTM. Ці результати підтверджують гіпотезу, що не модифікований метод мав не достатньо великої кількості інформації, а при наданні лише узагальненого тексту повнота контексту документа нівелювалась На рисунку 3.9 зображено схематичне представлення входу GPT-2 моделі.

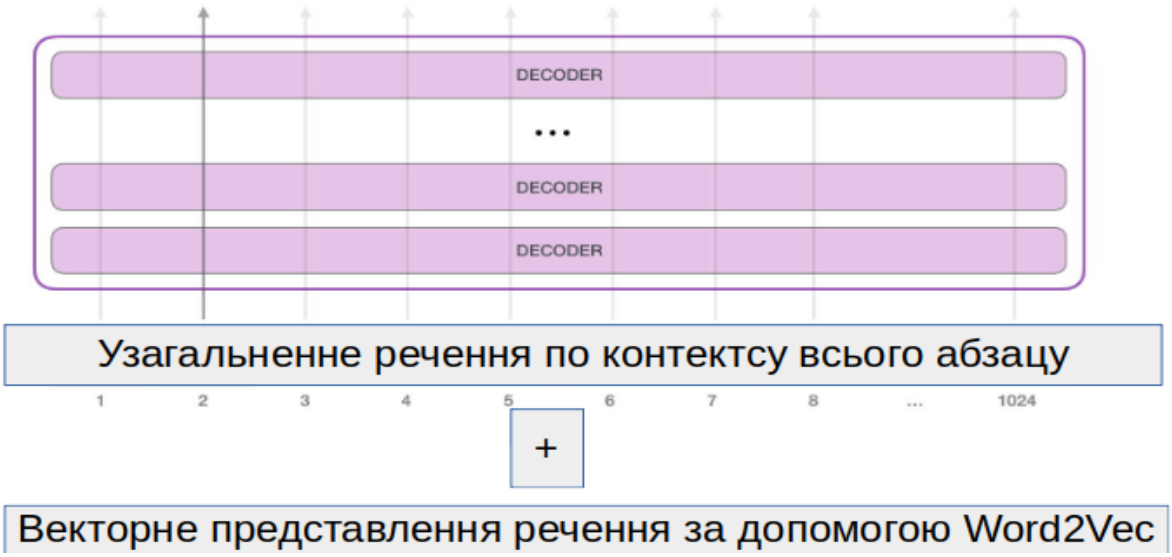

Рисунок 3.9 - Робота GPT-2 та вигляд даних, які потрапляють на вхід в покращеному методі

Попередні досліди було проведено лише з одною із LSTM моделей, а саме з трьох шаровою LSTM моделлю з мінімальною кількістю вхідних слів в 10. Але отримані результати по роботі не модифікованого метода є близькими один до одного ,тому логічним буде повторити дослід та отримати результати з різними наборами параметрів для модифікованого метода.

В рисунку 3.9 було показано схематичний вхід GPT-2 моделі. В якому узагальнене речення по приймається на вхід нейронної мережі разом із векторнем представлення абзацу текста. Векторне представлення абзацу текста було сформаване LSTM 3-шаровою моделлю. Яка беручи до уваги весь абзац, чи весь документ в залежності від параметрів, формує абстрактне узагальнення, яке потім являється додатковим контекстом для GPT-2 мережі при формуванні класифікації.

В таблиці 3.6 представлені результати проведених дослідження, в комбінації методів LSTM та GPT-2 різними параметрами для модифікованого метода.

Отримані результати свідчать про те, що основним параметром, який покращує якість класифікації, є розмірність GPT-2 моделі, та як наслідок кількість параметрів моделі. Також є різниця між кількістю шарів моделі, яка узагальнює дані, та якістю класифікації, але вона все ж менш відчутна. Залежність від мінімальної кількості слів в реченні є найменш чутливим параметром. По своїй суті, мінімальна кількість слів - це порог при якому метод узагальнення даних не бере до уваги речення. Зазвичай контекст такого речення значно менший ніж речення із більшою кількість слів. А при необхідності класифікації саме цього речення чи абзаца, дані про це речення, яке не було узагальнене потрапить все одно в GPT-2.

Таблиця 3.6 - Результати дослідження, в комбінації методів LSTM та GPT-2 різними параметрами для модифікованого метода

Розмірність LSTM-моделі	Мінімальна кількість слів в одному реченні	Кількість шарів Decoder в GPT-2	Розмір попередньо навченої GPT-2 моделі	Результат на тестовому наборі даних	Результат на навчальному наборі даних
LSTM 1layer	4	10	117M param	77.1%	80.2%
LSTM 1layer	4	30	345M param	81.0%	83.1%
LSTM 1layer	4	50	762M param	83.3%	86.3%
LSTM 2layer	7	10	117M param	79.1%	83.6%
LSTM 2layer	7	30	345M param	82.1%	87.1%
LSTM 2layer	7	50	762M param	83.4%	87.1%
LSTM 3layer	10	10	117M param	79.1%	82.1%
LSTM 3layer	10	30	345M param	82.0%	86.1%
LSTM 3layer	10	50	762M param	84.2%	88.6%

Проведемо порівняння класичного Баєсівого класифікатора та модифікованого GPT-2 + LSTM , що наведено в таблиці 3.7.

Таблиця 3.7 - Порівняння класичного Баєсевського класифікатора та модифікованого GPT-2 + LSTM

	Результат на тестовому наборі даних	Результат на навчальному наборі даних
Баєсевський класифікатор	76.1%	78.1%
GPT-2	82.1%	84.4%
GPT-2 + LSTM	67.1%	68.4%
GPT-2 + LSTM модифікований	84.2%	86.4%
LSTM«1layer»+GPT-2 «117M param»	77.1%	80.2%
LSTM«2layer»+GPT-2 «345M param»	81.0%	83.1%
LSTM«3layer»+GPT-2 «345M param»	82.0%	86.1%
LSTM«3layer»+GPT-2 «762M param»	84.2%	88.6%

Отриманий результат варто трактувати як те, що використання модифікованої комбінації GPT-2 та LSTM в будь-якому випадку буде краще для даного набору даних, ніж Баєсевський класифікатор. Також ці результати свідчать про те, що не модифікований алгоритм значно програє навіть методу Баєсевського класифікатора.

Висновки до третього розділу

В даному розділі розроблено та проаналізовано декілька методик. Дослідження показали, що на використаному наборі даних модифікований алгоритм LSTM+ GPT-2 являється найкращим. Було проведено пошук найкращих параметрів та запропонована модифікація початкового метода. Також проведене порівняння із класичними методами показало, що запропонований метод найкращий на заданому набору даних. Виявлені проблеми при формуванні початкового (не модифікованого) метода. Вирішення цих проблем дозволили впевнено покращити результати класифікації. Проте залишилась можливість модифікації метода, для покращення результату класифікацій. А саме, модифікація частини із узагальненням даних, додаванні додаткових параметрів в навчальну вибірку, таких як заголовок, та інше. Це дозволить більш якісно навчити та узагальнювати текст, як наслідок більш якісно буде сформований вектор-контекст, який допоможе GPT-2 класифікувати дані.

4 ВПРОВАДЖЕННЯ МЕТОДІВ В РЕАЛЬНІ ЗАДАЧІ, КЛАСИФІКАЦІЯ МЕТОДІВ

4.1 Класифікація методик за типом задач які вони вирішують

Різні типи узагальнення можуть бути корисними в різних задачах, тому можна класифікувати їх базуючись на типах задач, які ці методи призвані вирішувати. Методи узагальнення можна класифікувати на наступні категорії, це:

1) Залежно від типу деталей

Заснований на типі детального резюме може бути як інформативним, так і орієнтовними, рисунок 4.1. Орієнтовний короткий зміст використовується для швидкого перегляду довгого документа і містить лише основну ідею оригінального тексту. Вони, як правило, невеликі, і це спонукає користувача прочитати оригінальний документ. Наприклад, купуючи будь-який роман, покупець читає резюме, подане на зворотному боці роману.Інформаційний резюме служить заміною оригінального документа. Він надає стисну інформацію про оригінальний документ користувачеві.

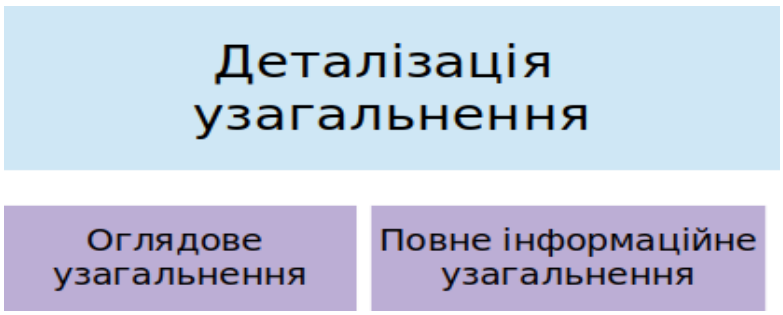

Рисунок 4.1 - Схема узагальнення за деталізацією

2) Залежно від змісту узагальнення

Можна узагальнювати документ на його оригінальній основі і не використовувати тематику документа, рисунок 4.2. А можна формувати узагальнювати за певною тематикою. Це забезпечує додаткову якість

узагальнення якщо необхідно із обширного оригінального документу знайти певний абзац, який стосується чогось конкретного.

Рисунок 4.2 - Схема узагальнення за певною тематикою

3) На основі залежності від жанрового типу

Тут можна розділити такі методики на два типи жанрово залежні методики, це методики які розроблені спеціально для обробки (газет, журналів, документів) вони добре узагальнюють лише той тип даних на якому навчені. І мають деякі особливості.

Жанрово незалежні методики, ці методики навчені на різному жанровому наборі даних. Вони в середньому кращі для узагальнення тексту який не має жанрової залежності (твіти, листи, звертання тощо), рисунок 4.3.

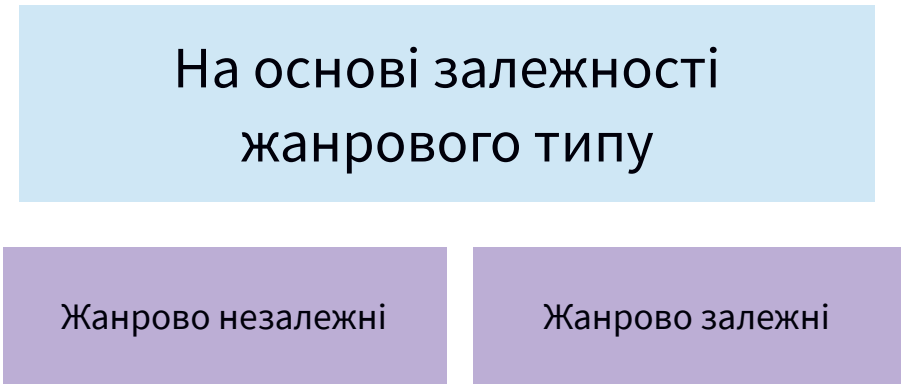

Рисунок 4.3 - Схема узагальнення за жанровим типом

4) На основі кількості вхідних документів

Узагальнення можна класифікувати на основі того скільки документів може приймати за один раз, рисунок 4.4.. Існують методики які дозволяють узагальнювати декілька документів, які зв’язані між собою одною тематикою. Та системи, які можуть приймати на вхід тільки один документ

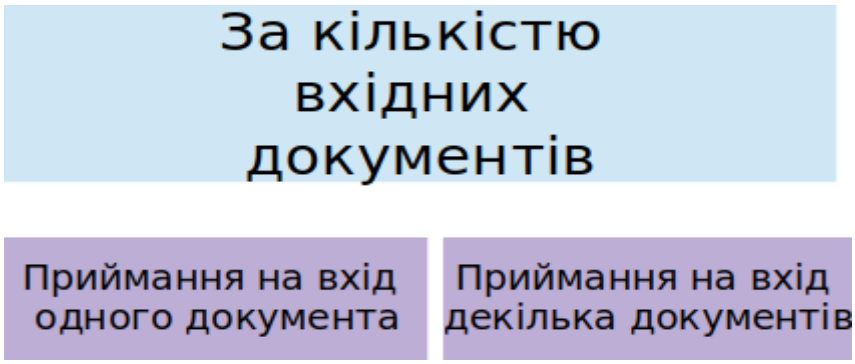

Рисунок 4.4 - Схема узагальнення за кількістю вхідних документів

5) На основі кількості мов якими може буде поданий документ

Існує одномовна методика, яка приймає та узагальнює тільки документи на одній мові, це мова на якій відбувалось навчання нейронної мережі.

Та багатомовні методи які дозволяють навчати нейронну мережу на великих корпусах тексту які складені із декількох мов. Зазвичай це мають бути історично близькі мови(з одної мовної групи), рисунок 4.5.

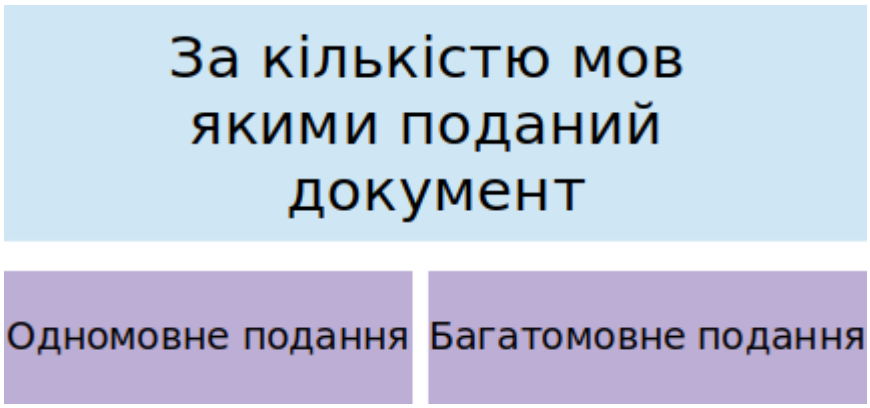

Рисунок 4.5 - Схема узагальнення за кількістю мов

4.2 Практична цінність, розгляд прикладів використання, класифікація власного метода.

Задача обробка інформації, яка представлена в природномовній формі, актуальна із давніх часів.

Дана робота присвячена дослідженню існуючих методів та розробці нових методів з класифікації даних, пошуку подібностей, а також проведенню класифікації методів обробки даних. Прикладами задач, які повинні вирішувати дана робота – це:

- пошук подібностей в тексті для виявлення плагіату покращення захисту інтелектуальної власності;
- класифікація текстів для створення адаптивного підбору новин, покращення алгоритмів рекомендацій, автозаповнення, підпошук текстів в великих документів.

Розроблений метод класифікації можна класифікувати за запропонованою класифікацією як одномовний, без використання тематик, жанрово залежний, однодокументний, із оглядовим узагальненням.

4.3 Застосування представленої комбінації методів в проєкті

«Microsoft Teams»— центр для командної роботи в Office 365 від Microsoft, який інтегрує користувачів, вміст і засоби, необхідні команді для ефективнішої роботи. Додаток об'єднує все в спільному робочому середовищі, яке містить чат для нарад, файлообмінник та корпоративні програми. Розроблений для смартфонів, що працюють на платформах Android, IOS, Windows Phone і комп'ютерів з операційною системою Windows та дистрибутивів Linux.

Microsoft Teams є конкурентом таких сервісів, як Slack, і є еволюційним оновленнями від Microsoft Skype для бізнесу, рисунок 4.6.

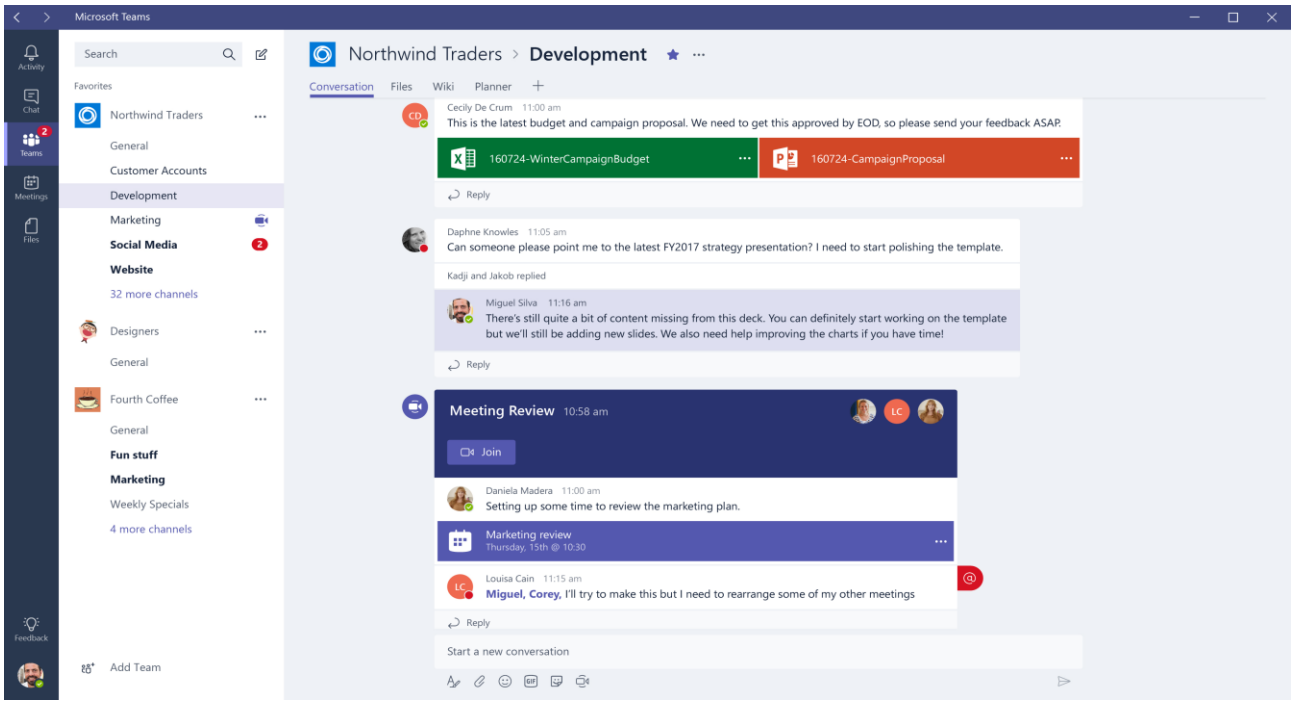

Рисунок 4.6-Microsoft Teams

Віртуальний асистент (англ. Virtual assistant) - програмний агент, який може виконувати завдання (або сервіси) для користувача на основі інформації, введеної користувачем, даних про його місцезнаходження, а також інформації, отриманої з різних інтернет-ресурсів (погода, вуличний рух, новини, курси валют і цінних паперів, роздрібні ціни в магазинах і т. д.). Прикладами такого роду агентів є програми Siri, Google Assistant (Google Now), Amazon Alexa, Microsoft Cortana, Bixby, Voice Mate, Аліса і інші.

Створення основної вебчастини лягло на Bot Framework. Платформа Bot Framework, а також служба Azure Bot надають кошти для створення, тестування, розгортання і управління інтелектуальними програми-роботиамі в одному місці. The Bot Framework включає модульний і розширюваний пакет SDK для створення програми-роботи, а також інструменти, шаблони і пов'язані служби штучного інтелекту. За допомогою цієї платформи розробники можуть створювати програми-роботи, які використовують мову, знання природної мови, опрацювання питань і відповідей і багато іншого.

Боти забезпечують взаємодію, більше схожу не на роботу з комп'ютером, а на спілкування з живою людиною, ну або хоча б з дуже розумним роботом. Вони допоможуть перенести на автоматизовані системи прості та повторювані завдання, такі як резервування столиків в ресторані або збір відомостей для профілю без необхідності прямої участі людини. Користувачі взаємодіють з ботом, використовуючи текстові повідомлення, інтерактивні карти і мова. Взаємодія з ботом може обмежуватися простими питаннями і відповідями або являти собою складне інтелектуальне спілкування з наданням доступу до служб.

При розробці програмного забезпечення, використовувалась мова програмування Python. Дана програма представляє собою «Timekeeper» -бот асистент, інтелектуальний помічник. Який є комплексом програмного забезпечення, ресурсів і бібліотек. Розроблений для автоматизації(оптимізації) процесів контролю, та адміністрування розробки ПО (та інших задач).

Направлений на максимально схожу взаємодію з користувачем як інший користувач, імітує поведінку людини. Бот асистент підтримує спілкування в тестовому, голосовому та графічному режимах. Обробляє, та створює звіти за запитом користувача. Веде журнали роботи, а також планування задач. Сповіщає користувача про поставлені задачі, і веде подальший контроль поставлених задач. Має функціонал автокорекції помилок тексту запитів спілкування, а також автокорекції задач.

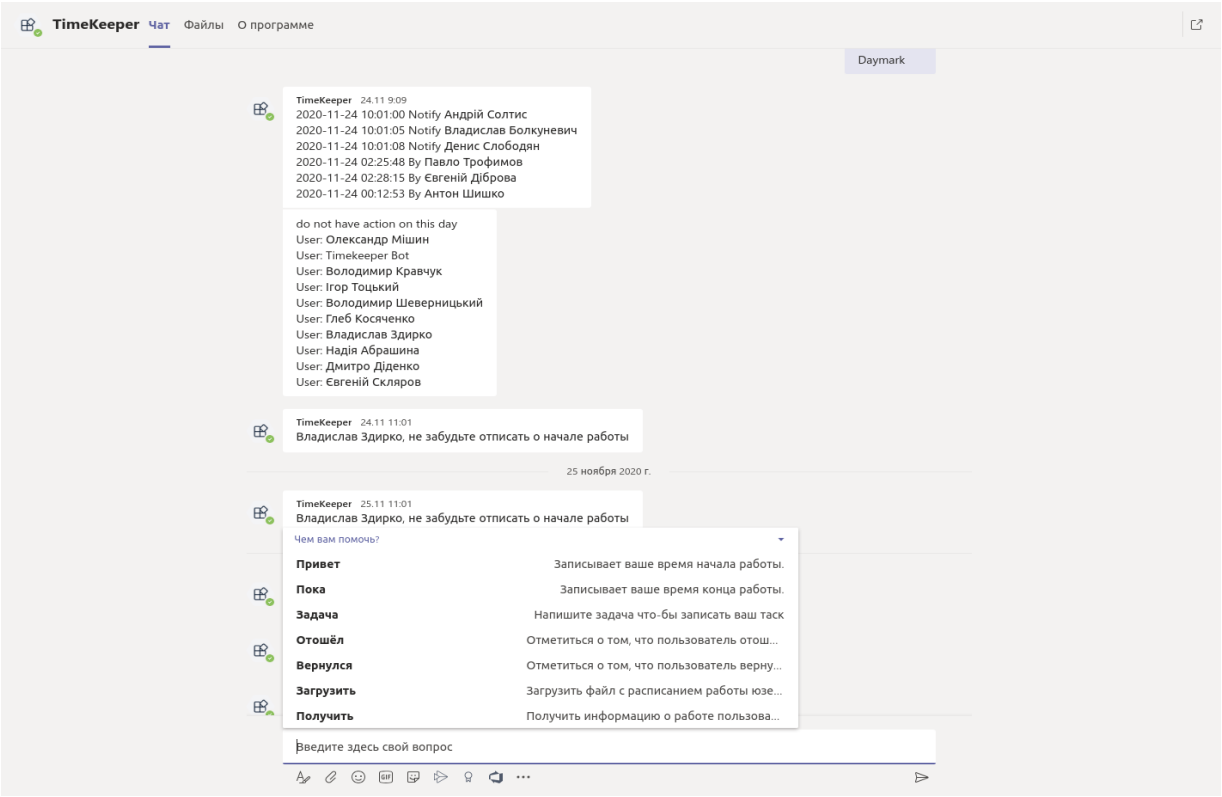

Рисунок 4.7-Приклад інтерфейсу бота помічника в середовищі Teams

Програми-роботи - це дуже схоже на сучасні веб-додатки, які живуть в Інтернеті і використовують API-інтерфейси для відправки та отримання повідомлень, рисунок 4.3. Вміст бота може бути найрізноманітнішим залежно від його типу і призначення. Програмне забезпечення сучасних ботів спирається на складний набір технологій і засобів, що дозволяють надавати все більш складні можливості на широкому спектрі платформ. Але можуть

існувати і найпростіші боти, які вміють лише отримувати текстове повідомлення і повертати його користувачеві.

Боти можуть виконувати всі ті ж дії, що і інші види програмного забезпечення: читати і зберігати файли, використовувати інтерфейси API і бази даних, робити обчислення. Унікальність ботів полягає в тому, що крім цього вони використовують механізми, традиційно задіяні для обміну даними між людьми, рисунок 4.8.

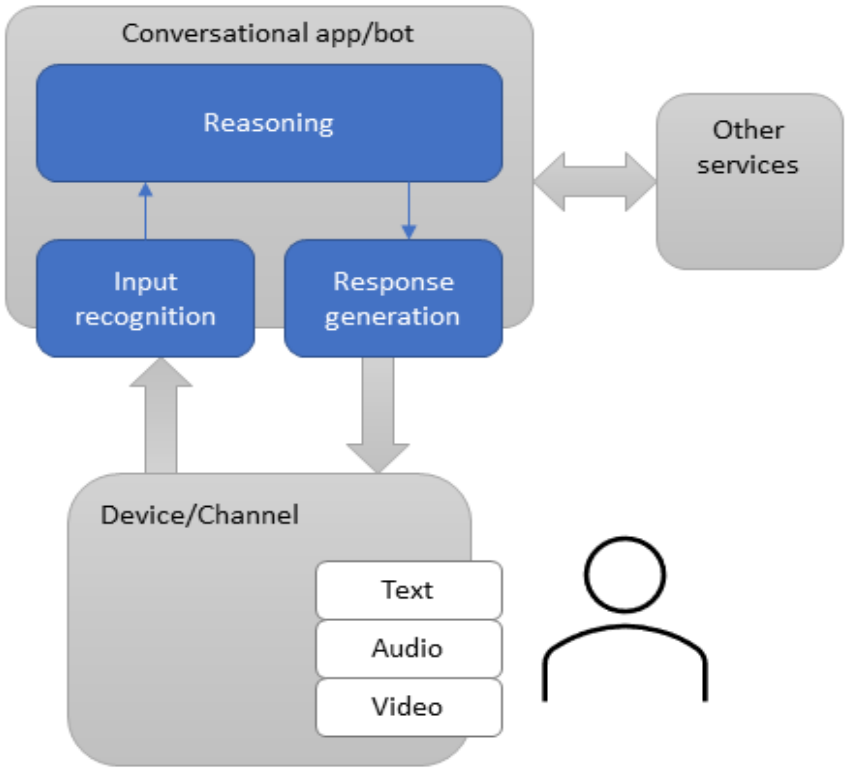

Рисунок 4.8 - Схема взаємодії ресурсів та користувача

Іноді Bot повинен звертатися до захищених мережевих ресурсів від імені користувача, наприклад перевіряти електронну пошту, перевіряти стан рейсу або розміщувати замовлення. Користувач повинен авторизувати програму-робота зробити це від свого імені і, щоб авторизувати робота, користувач повинен пройти перевірку справжності. OAuth використовується для перевірки автентичності користувача та авторизації робота. Також типи

перевірки аутентичності. На наступному рисунку 4.9 показані елементи, які беруть участь в процесі аутентифікації.

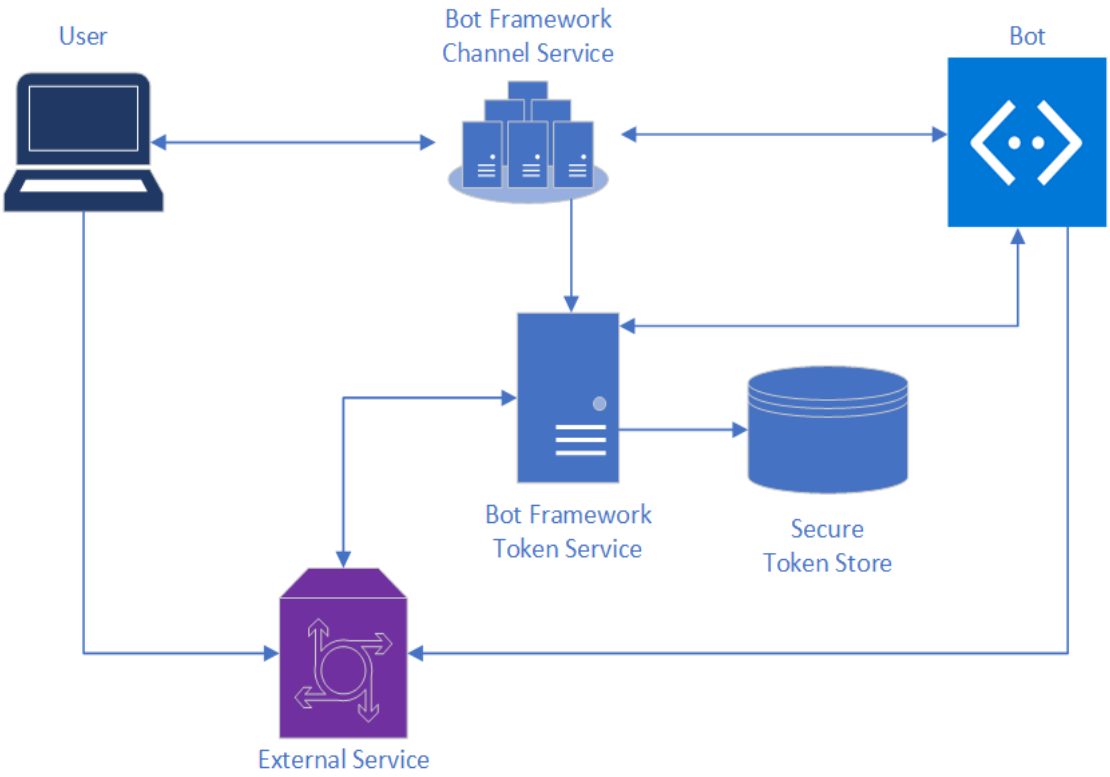

Рисунок 4.9 - Схема процесу аутентифікації.

Основними задачами бота є автокорекція тексту, та планування задач, обробки та створення журналів відвідувань, та звітів роботи. Оскільки основні проблеми створення бота та його налаштування не є частиною даної тематики роботи, то буде розглянуто основні задачі. Прикладом необхідних дій які бот успішно виконує це класифікація тексту.

Розглянемо на прикладах, які вирішує бот. При неправильній постановці завдання вести структурований пошук задач та налаштування робочого процесу затрудняється. Тому даний метод дозволяє при невеликому навчанні визначати фрази в тексті та класифікувати їх як ключові речення. Приклад вирішення задачі

TimeKeeper Привет

Вчера и сегодня до вечера был занят, сейчас доделываю задачи. Смержил ветки, сейчас доделываю классы (наследование интерфейсов и реализация), и пишу сериализаторы.
Интерактивное обучение

TimeKeeper Привет

Вчера и сегодня до вечера был занят, сейчас доделываю задачи. Смержил ветки, сейчас доделываю классы (наследование интерфейсов и реализация), и пишу сериализаторы.

Проект: Интерактивное обучение

Майлстоун:-

В даному прикладі алгоритм виявив «Интерактивное обучение» як найменування проекта над яким ведеться робота, класифікувавши задане речення як підпис до проекта або задачі. Осклільки при створенні навчального набору було визначено що задача завжди описана із url-адресою. То алгоритм підстановки текстів отримавши дані про надлежність цього речення до певного типу зробив допис «Проект». Це дозволить в подальшому при пошуку підзадач легко виявити задачу яка відноситься до «Проект: Интерактивное обучение».

Наступною задачею автокорекція використовується звичайний порівняння слів за їх векторною формую, таким чином векторно близькі слова привітання які визначають початок роботи (Привіт, Хай, Привет, Доброго Дня, Good Morning? Hello, Ні та інші.) Будуть векторно близькими, і таким же чином ми матимемо автокорекцією слова. В ситуації коли букви не правильні, бот порівнює слова по їх довжині з словами із заданого словника, а далі якщо знаходить відповідність заміняє на также по семантичному значенні слово з

помилкою на нове слово без помилки із таким же або краще відповідне до цього контексту слово. Таким чином, записує в журнал правильні сформовані речення, рисунок 4.10.

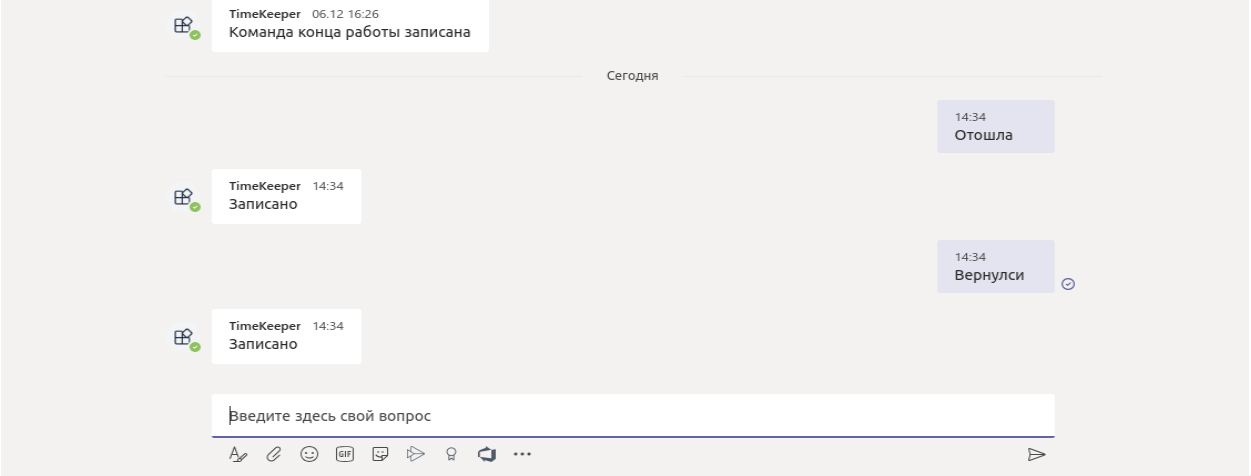

Рисунок 4.10 - Автокорекція та незважання на помилки

На жаль можливості корегування відправленого повідомлення в середині Teams, зі сторони бота не є можливим. Проте в внутрішньому БД сформовані повідомлення без помилки. Тому при подальшому пошуку дій користувача, та створення журналів дані повідомлення будуть відображатись коректно.

Отже цей бот-асистент завдяки методам машинного навчання значно полегшує роботу адміністрування процесів створення програмного забезпечення та іншого. В даному боті було успішно використана створені методи по класифікації даних.

Висновки до четвертого розділу

Показано та доведено, що з покращенням розуміння природної мови комп’ютери зможуть вчитися з інформації в режимі он-лайн та застосовувати те, що вони дізналися, у реальному світі. У поєднанні з генерацією природних мов комп’ютери стануть дедалі більше здатними приймати та давати вказівки. Доведено, що через швидке зростання технологій та використання Інтернету відбувається інформаційне перевантаження. Цю проблему можна вирішити, якщо є методи текстового узагальнювання, які дають короткий зміст документ для допомоги користувачеві. Отже, існує потреба в розробці системи, де користувач може ефективно отримувати та отримувати узагальнений документ. В даному розділі було запропонована класифікація методів узагальнення та класифікації, а також успішно застосовано створені методи по класифікації даних. Перевірений на реальних задачах запропонований метод довів свою корисність, в порівнянні із іншими методами шанси успішно класифікувати текст вище чим у звичайних класичних методів.

ЗАГАЛЬНІ ВИСНОВИКИ ПО РОБОТІ

У даній роботі було розроблено метод класифікації даних із залученням узагальнення. Було досліджено пошук оптимальних параметрів, та наведені обґрунтування використання даного метода. Було проведено порівняння методів класифікації.

Під час вирішення поставлених задач по створені нового метода були такі складнощі як, перенавчання машинної мережі, неякісні дані. Для їх вирішення було сформовано модифікацію яка дозволила при незначній зміні метода значно покращити його результативність в класифікації текстів.

Як логічне продовження створення методів було застосовано отриманих результатів на практичних завданнях. А саме створення бота асистента, інтелектуального помічника. Який є комплексом програмного забезпечення, ресурсів і бібліотек. Розроблений для автоматизації(оптимізації) процесів контролю, та адміністрування розробки ПО(та інших задач). Направлений на максимально схожу взаємодію з користувачем як інший користувач, імітує поведінку людини. Бот асистент підтримує спілкування в тестовому, голосовому та графічному режимах. Обробляє, та створює звіти за запитом користувача. Веде журнали роботи, а також планування задач. Сповіщає користувача про поставлені задачі, і веде подальший контроль поставлених задач. Має функціонал автокорекції помилок тексту запитів спілкування, а також автокорекції задач.

Одною із важливих частин є створення класифікації методів узагальнення та класифікації машинного навчання. Створена класифікація дозволяє швидко шукати та групувати методи, було проведено класифікацію власного метода на основі створеної класифікації. Класифікація буде корисна для вирішення практичних задач, при пошуку найкращого алгоритму для конкретної задачі. А також в формуванні чітких пунктів, які повинна вирішувати нейронна мережа для вирішення поставленого перед нею завдання.

В якості майбутніх напрямків досліджень та покращень методів вибрано наступні.

Модифікація створених методів, а саме модифікація частини узагальнення. Особливу увагу варто приділити в попередній обробці даних, в сучасності де існують великі набори даних вони не є адаптованими під конкретні випадки. Використовуючи попередню оцінку даних, необхідно особливо звернути увагу на збалансування набору даних та використання екстрактивних методів узагальнення для створення нових методик класифікації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1.Cheng, Jianpeng, and Mirella Lapata. "Neural summarization by extracting sentencesand word. 2018.

2. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781

3.Chandrasekar, R. Doran, C. and Srinivas, B., Motivations and Methods for Text Simplification, 1996.

4.Neural Word Embedding as Implicit Matrix Factorization Omer Levy and Yoav Goldberg 2014

5.Aakash Sinha, Abhishek Yadav, Akshay Gahlots Extractive Text Summarization using Neural Networks. 2018.

6.Alexander M. Rush, Sumit Chopra, Jason Weston A Neural Attention Modelfor Abstractive Sentence Summarization. 2015.

7.Josef Steinberger and Karel Jezek Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. 2003.

8.Cheng, Jianpeng, and Mirella Lapata. "Neural summarization by extracting sentencesand word. 2018.

9.Freitas, N., and A. Kaestner. "Automatic text summarization using a machine learning approach." Brazilian Symposium on Artificial Intelligence (SBIA), Brazil. 2005.

10. Aakash Sinha,Abhishek Yadav,Akshay Gahlot. Extractive Text Summarization using Neural Networks. 2008.

11. Bag of Tricks for Efficient Text Classification [Електронний ресурс] / A.Joulin, E. Grave, P. Bojanowski, T. Mikolov –Режим доступу до ресурсу: https://arxiv.org/pdf/1607.01759.pdfi

12.Domingos, Pedro & Michael Pazzani (1997) «On the optimality of the simple Bayesian classifier under zero-one loss».*Machine Learning*, 29:103-137.

13.McCallum, A. and Nigam K. «A Comparison of Event Models for Naive Bayes Text Classification».

14.Bhadeshia H. K. D. H. (1999).Neural Networks in Materials Science.*ISIJ International* **39**(10): 966–979.

15.1944-, Smith, Murray, (1993).*Neural networks for statistical modeling*. Van Nostrand Reinhold.

16.C.L. Giles, C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun, Y.C. Lee,«Learning and Extracting Finite State Automata with Second-Order Recurrent Neural Networks», Neural Computation,